НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги»

Виконала:
студентка IV курсу, групи КП-52
Левошко Катерина Василівна _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.
Люшенко Л.А. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.ф.-м.н., доц.
Шубенкова І.А. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

«ЗАТВЕРДЖУЮ»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студентці
Левошко Катерині Василівні

1. **Тема проекту:** «Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги», керівник проекту Люшенко Леся Анатоліївна, к.т.н., старший викладач, затверджена наказом по університету № 1331-С від «22» травня 2019 р.
2. **Термін подання** студентом завершеного проекту: «19» червня 2019 р.
3. **Вихідні дані для дипломного проектування:** див. Технічне завдання.
4. **Зміст пояснювальної записки:**
   – аналіз існуючих рішень та актуальність теми;
   – обґрунтування вибору засобів реалізації;
   – розроблення мобільного додатка;
   – аналіз розробленої системи.
5. **Перелік обов'язкового ілюстративного матеріалу:**
   – алгоритм взаємодії графічних елементів (креслення);
   – схема бази даних (креслення);
   – діаграма варіантів використання (плакат);
   – дерево проблем системи (плакат).
6. **Консультанти розділів проекту**

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. **Дата видачі завдання: «**31» жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

№ з/п	Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання	Термін виконання	Примітка
1.	Вивчення літератури за тематикою проекту	07.11.2018	
2.	Розроблення та узгодження технічного завдання	20.11.2018	
3.	Розроблення структури системи	10.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	21.12.2018	
5.	Розроблення дизайну екранів та графічних елементів	28.01.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	15.02.2019	
7.	Програмна реалізація системи	15.03.2019	
8.	Тестування системи	22.03.2019	
9.	Підготовка матеріалів третього та четвертого розділу дипломного проекту	15.04.2019	
10.	Підготовка графічної частини дипломного проекту	10.05.2019	
11.	Оформлення документації дипломного проекту	01.06.2019	

Керівник дипломного проекту _____ Л.А. Люшенко

Студент _____ К.В. Левошко

АНОТАЦІЯ

Цей дипломний проект присвячений створенню програмної системи планування медичного обслуговування у лікарів первинної ланки медичної допомоги.

Програмна система являє собою мобільний додаток, який містить статичні та динамічні сторінки. Статичні сторінки носять інформаційний характер для користувача. Завдяки їм користувач отримує інформацію про наявних лікарів, актуальні медичні статі та необхідні дані про мобільний додаток. Динамічні сторінки взаємодіють з календарем, на основі якого побудована логіка програмного застосунку.

Динамічна частина мобільного додатку забезпечує доступ зареєстрованих користувачів до календаря з подіями цих користувачів. Користувачі можуть вносити записи про лікарські засоби, формувати автоматичний графік їх прийому, налаштовувати сповіщення про прийом лікарських засобів, запис до лікаря та записи про стан самопочуття. А також відслідковувати історії захворювань.

Інформаційна безпека програмного додатку реалізована за рахунок надання прав доступу тільки зареєстрованим користувачам. Зареєстровані та авторизовані користувачі мають доступ до статичних та динамічних сторінок програмного додатку та можуть змінювати інформацію у динамічній частині, в залежності від їхніх потреб.

У цьому дипломному проекті розроблено: архітектуру програмної системи, алгоритм взаємодії графічних елементів, процедуру створення нагадувань користувачем, а також графічні елементи та дизайн мобільного додатку.

ABSTRACT

This diploma project deals with the development of the programmed system of medical planning for primary care physicians.

The software system is a mobile application that contains static and dynamic pages. Static pages are informational for the user. Thanks to them, the user receives information about actual doctors, relevant medical articles and the necessary data about the mobile application. Dynamic pages interact with the calendar, on the basis of which the logic of the software application is built.

The dynamic part of the mobile application provides registered users access to the calendar with the events of these users. Users can enter medical records, create an automatic schedule for taking them, set up notifications about taking medications, recording to a doctor and recording a state of health. And also track the history of diseases.

Information security of the software application is realized by granting access rights only to registered users. Registered and authorized users have access to the static and dynamic pages of the software application and can change the information in the dynamic part, depending on their needs.

The following structures and algorithms are developed in this project: the architecture of the software system, the algorithm of interaction of graphic elements, the procedure for creating reminders by the user, as well as graphic elements and the design of the mobile application.

ДП.045440-01-90 Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмна система	5	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	Технічне завдання		
ДП.045440-03-81	Програмна система	61	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	Пояснювальна записка		
ДП.045440-04-51	Програмна система	4	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмна система	22	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмна система	1	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	Алгоритм взаємодії		
	графічних елементів.		
	Схема алгоритму		
ДП.045440-07-99	Програмна система	1	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	База даних системи.		
	Схема даних.		
ДП.045440-08-98	Програмна система	1	
	планування медичного		
	обслуговування у лікарів		
	первинної ланки		
	медичної допомоги.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНА СИСТЕМА ПЛАНУВАННЯ МЕДИЧНОГО ОБСЛУГОВУВАННЯ У ЛІКАРІВ ПЕРВИННОЇ ЛАНКИ МЕДИЧНОЇ ДОПОМОГИ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ К.В. Левошко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання сучасними людьми в якості інформаційного забезпечення в сфері медичного обслуговування з метою надання користувачеві можливості контролювати власне здоров'я та планувати медичне обслуговування.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмна система планування медичного обслуговування повинна забезпечувати наступні основні функції:

– можливість додавання заміток про прийом лікарських засобів;

– можливість додавання заміток про прийом у лікаря;

– можливість додавання заміток про самопочуття;

– можливість створювати графік прийому лікарських засобів;

– перегляд медичних тематичних статей;

– створювати та зберігати історії захворювання.

Розробку виконати на платформі iOS (мова програмування Swift) з використанням шаблону проектування MVC.

Додаткові вимоги:

– наявність меню програмного додатка;

– наявність логотипу та назви програмного додатку;

– наявність анімованих кнопок;

– наявність згоди на обробку персональних даних та користувацької згоди;

– дизайн сторінок з використанням білого, блакитного та синьо-зеленого кольорів.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

– «Алгоритм взаємодії графічних елементів»;

– «Схема бази даних».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи 07.11.2018

Розроблення та узгодження технічного завдання 20.11.2018

Розроблення структури системи..... 10.12.2018

Розроблення дизайну екранів та графічних елементів..... 28.01.2019

Програмна реалізація системи..... 15.03.2019

Тестування системи..... 22.03.2019

Підготовка матеріалів текстової частини проекту 15.04.2019

Підготовка матеріалів графічної частини проекту..... 10.05.2019

Оформлення технічної документації проекту 01.06.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ПЛАНУВАННЯ МЕДИЧНОГО ОБСЛУГОВУВАННЯ У ЛІКАРІВ ПЕРВИННОЇ ЛАНКИ МЕДИЧНОЇ ДОПОМОГИ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ К.В. Левошко

2019

ЗМІСТ

ВСТУП

Популярність цифрових технологій виросла в десятки разів. Зараз покриття мережі Інтернет присутнє у всіх великих містах світу. Кількість користувачів мережі Інтернет складає половину від всього населення планети станом на 2017 рік. А в розвинених країнах – 81% [1]. З розвитком технологій всі види діяльності, в тому числі і підприємницька, почали масштабуватись. За останнє десятиріччя більша частина видів діяльності інтегрується у напрямку цифрових технологій.

Медична сфера не є винятком, вона активно розвивається в цифровому світі сьогодні. Через високий попит людей до контрою свого здоров'я, з'являються програмні додатки для відслідковування показників організму. В стані повного інформаційного навантаження сучасна людина потребує систему для контролю медичного обслуговування. Також, вимогами до програмного рішення є доступність на різних платформах, як веб-ресурс, так і мобільний додаток. Тож, розробка програмного рішення у сфері медицини є актуальним рішенням на сьогоднішній день.

Проект, який розроблюється в рамках бакалаврської дипломної роботи, створюється для контролю власного здоров'я сучасними людьми. Програмне рішення буде представлено у вигляді мобільного додатка для звичайних користувачів. Програмний продукт надає можливості для ефективного слідкування за здоров'ям та допомагає користувачеві пам'ятати про необхідні медичні процедури. Користувач повинен лише мати смартфон та доступ в мережу Інтернет.

Сучасна людина намагається вирішити всі можливі рутинні справи свого життя через мережу Інтернет. За останні декілька років основним інструментом для зв'язку з цифровим світом є смартфон. Згідно статистики за 2017 рік в Україні майже 70% населення від 18 до 50 років користуються смартфонами. А 91% від респондентів використовують мобільні додатки на

смартфонах. За оцінками аналітиків до 2020 року цей показник перевищить відмітку 70% [2].

Таким чином, розробка мобільного додатка в сфері медичного обслуговування є актуальним рішенням на сьогоднішній день з перспективою розвитку.

МЕТА ДИПЛОМНОГО ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧІ

Метою дипломного проекту є розробка програмного додатка планування медичного обслуговування у лікарів первинної ланки медичної допомоги.

Згідно до мети дипломного проекту було поставлено наступні задачі:

- визначення актуальності предметної області та проблеми, що вирішує програмний додаток, огляд існуючих рішень поставленої проблеми та проведення аналізу цих програмних додатків шляхом порівняння переваг та недоліків, огляд та аналіз сучасних технологій, сформування вимог для розробки програмного додатка;
- вибір засобів реалізації мобільного додатка, аналіз існуючих платформ, мов програмування, бібліотек, технологій, середовищ розробки;
- опис архітектури системи, функціональних вимог до системи, особливості реалізації шаблонів проектування та розробка програмного забезпечення;
- тестування готового мобільного додатка, аналіз вимог програмного забезпечення, контроль якості програмного продукту.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

TLS/SSL – Transport Layer Security (захист на транспортному рівні передачі даних), Secure Sockets Layer (рівень захищених сокетів);

MVC – Model-View-Controller (архітектурний шаблон «Модель-Представлення-Контролер»);

SCRUM – підхід управління проектами для гнучкої розробки програмного забезпечення. Scrum чітко робить акцент на якісному контролі процесу розробки;

iOS – iPhone operating system;

macOS – Macintosh Operating System

IDE – Integrated Development Environment;

SSH – Secure Shell;

JVM – Java Virtual Machine;

SDK – software Development Kit;

REPL – read-eval-print loop;

PHP – Hypertext Preprocessor;

JS – Java Script;

Java SE – Java Standart Edition;

JSON – JavaScript Object Notation

SQL – Structured Query Language;

ACID – Atomicity, Consistency, Isolation, Durability;

SSL – Secure Sockets Layer;

GUI – Graphical user interface

HTML – Hypertext Markup Language

QA – Quality Assurance;

ПЗ – програмне забезпечення;

АЗ – апаратне забезпечення;

РФ – Російська Федерація;

ОС – операційна система;

ООП – об’єктно-орієнтовне програмування;

БД – база даних;

СУБД – система керування базами даних.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА АКТУАЛЬНІСТЬ ТЕМИ

1.1. Актуальність систем контролю здоров’я та постановка проблеми

На сьогоднішній день набрали популярності програмні додатки для смартфонів з великою кількістю функцій відстеження показників організму людини. Для цих пристроїв було створено окремі операційні системи – Android Wear (Google) та Watch OS (Apple). В основу Android Wear лягла технологія стеження за здоров’ям [3], включаючи фітнес-активність власника пристрою. Згідно статистики [4] людина не проти мати програмний додаток, що буде допомагати контролювати своє здоров’я. Можна створити програмний додаток, який буде працювати з людиною у інтерактивному режимі, а не представляти набір числових даних.

Майже всі люди відвідують лікаря, коли вже сталась проблема зі здоров’ям, і вони хочуть виправити цю проблему. Але більшість з них забуває про профілактичні огляди, які вони повинні відвідувати певну кількість разів в залежності від галузі. Наприклад, відвідування стоматолога повинно відбуватись два рази на рік, відвідування гінеколога (для жінок) два рази на рік, робити флюорографічне дослідження раз на рік (для своєчасного виявлення туберкульозу або раку легень) і т. д [5]. Існує ряд правил, які кожна людина повинна виконувати для контролю свого здоров’я.

Опис основних проблем з контролем здоров’я можна побачити на дереві на рис. у вигляді ієрархічної структури.

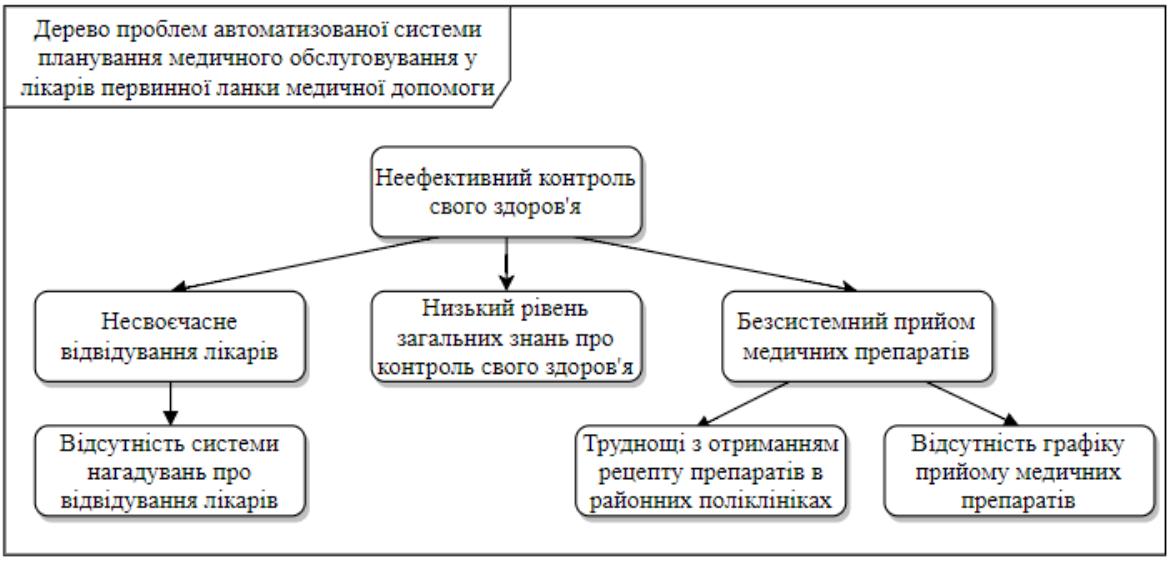


Рис. 1. Дерево проблем з контролем здоров’я в Україні

Неефективний контроль свого здоров’я може негативно вплинути на якість та тривалість подальшого життя, а саме:

- несвоєчасне відвідування лікарів – людський фактор, що проявляється в байдужому ставленні до власного здоров’я, відвідування лікарів, коли проблема сталась, а не коли її можливо передбачити. Проблема, яку зможе розв’язати розроблюване ПЗ –це відсутність системи нагадувань про відвідування лікарів;
- низький рівень загальних знань про контроль свого здоров’я, тобто недостатня обізнаність переліку необхідних відвідувань лікаря протягом року та/або ставлення до цієї інформації як до непотрібної у школі;
- безсистемний прийом медичних препаратів – самолікування є основною причиною цієї проблеми, також труднощі з отриманням рецепту у районних поліклініках та відсутність графіку прийому медичних препаратів.

Умови для вирішення проблеми – наявність у користувача смартфона з ПЗ для контролю свого здоров’я.

1.2. Аналіз існуючих програмних додатків контролю здоров'я

1.2.1. *Мобільний додаток «Моё здоровье: образ жизни»*

Мобільний додаток «Моё здоровье: образ жизни» [6] призначений для контролю власного здоров'я. Програмний додаток функціонує для платформ Android та iOS. Після встановлення програмного додатка його треба додати в налаштування смартфону, щоб всі дані правильно синхронізувались. За основу додатка взято календар, на якому можна створювати різного роду події або нагадування. Містить щоденник, де можна фіксувати артеріальний тиск, частоту серцевих скорочень, глюкозу крові (контроль цукру в крові), масу тіла, зріст, температуру та інші показники здоров'я, такі як здоровий сон, спосіб життя.

Для жінок є можливість відстежувати менструальний цикл, введення вагітності та здоров'я жінки в цілому. Також є функція самостійного ведення медичної карти, тобто всі дані про здоров'я будуть зберігатись в одному додатку, що найважливіше безкоштовно, без реклами, без вбудованих покупок. Є можливість використання додатка для всієї родини, ведучи сімейний щоденник здоров'я. Організована система нагадувань, яку людина сама може налаштувати як їй буде зручно. Існує розділ «Щоденник прийому ліків», де представлено графік прийому ліків на основі призначень лікаря, нагадування про прийом ліків, а також доступний довідник лікарських препаратів: прийом ліків за розкладом та в правильних дозах.

За наданою від розробників інформацією всі розділи сформовані за участю практикуючих лікарів [7].

Цей програмний додаток буде корисно людям, у яких виявлено діагноз: цукровий діабет 1, 2 типу. Лікарі завжди рекомендують виміряти рівень цукру в крові, виміряти артеріальний тиск і записати показники. Причому робити так кожен раз, тобто вести щоденник діабетика. Програмний додаток допоможе нічого не упустити: буде вести графік

схуднення (зниження ваги), зберігати вимір пульсу і тиску. Рівень глюкози (цукор в крові) буде представлена на окремому графіку [7].

Переваги програмного додатка:

– можливість мати декілька акаунтів в одному додатка;
– містить функції відстеження показників репродуктивного здоров'я в залежності від статі (важлива функція особливо для жінок);
– використання надійного криптографічного протоколу TLS/SSL при передачі даних.

Недоліки програмного додатка:

– інтерфейс додатка створено без урахувань вимог до проектування інтерфейсу;
– немає можливості завантажити аналізи або інші документи в форматах .pdf, .docx для зберігання;
– немає можливості видалення щоденника вагітності, якщо жінка, яка використовує програмний додаток не вагітна;
– програмний додаток функціонує в межах Російської Федерації, отже приймаючи користувацьку згоду, громадянин України не зможе відстояти свої права в разі судових справ.

Цей програмний додаток з усіх переглянутих є найбільш вдалим вирішенням поставленої проблеми, але через ряд недоліків використання даного додатка є проблематичним.

1.2.2. *Мобільний додаток «Дневник здоровья»*

Мобільний додаток «Дневник здоровья» [8] призначений для контролю власного здоров'я. Програмний додаток функціонує для платформи Android. Після встановлення програмного додатка треба його додати в налаштування смартфону, щоб всі дані правильно синхронізувались. За основу програмного додатка взято таблицю, у якій можна відмічати вагу, розміри тіла, тиск, пульс, рівень цукру та

холестерину, а також три доступні колонки, які можна використовувати для вимірювання власних показників.

У складі програмного додатка є довідник референтних значень показників. Вільно конфігурується склад і вид таблиці для накопичення показників. Кількість, склад показників і їх формат обираються самостійно. Спочатку встановлюється набір показників характерний для цукрового діабетом. Ця програма найкращим чином підходить для людей, які страждають серцево-судинними захворюваннями, хронічною нирковою недостатністю, цукровим діабетом, зайвою вагою, гормональними порушеннями, і багатьох інших, де потрібен постійний періодичний контроль. А також просто для здорових людей, що займаються спортом і захоплених фітнесом. Історія значень життєвих показників здоров'я можуть виявитися корисними також для лікаря, який відштовхуючись від результатів спостережень, зробить правильні висновки про стан здоров'я людини.

Переваги програмного додатка:

– графічне відображення числових показників;

– реєстрації та зв'язку з поштою.

Недоліки програмного додатка:

– відсутність функцій створення подій та нагадувань, що є важливими функціями у вирішенні поставленої задачі.

Програмний додаток не розв'язує поставлену проблему, він лише допомагає частково у певному вирішенні. Також, цей програмний додаток має низьку оцінку в Play Market (3.2/5) та малу кількість завантажень (приблизно 1000).

1.2.3. *Мобільний додаток «Youwell»*

Мобільний додаток «Youwell» [9] призначений для контролю власного здоров'я. Програмний додаток функціонує для платформи iOS.

Після встановлення програмного додатка треба його додати в налаштування смартфону, щоб всі дані правильно синхронізувались.

Youwell – програмний додаток для турботи про здоров'я, який містить наступні функції:

- створення індивідуальних графіків прийому ліків і отримання нагадувань;
- створення графіків тиску, пульсу, ваги, рівня цукру за числовими показниками;
- збереження результатів лабораторних досліджень в спеціальних формах в додатка. За колірним індикаторами дуже легко визначити, чи в нормі показники і для цього не потрібні спеціальні медичні знання;

Переваги програмного додатка:

- наявність колірних індикаторів показників для визначення норми показників людини.

Недоліки програмного додатка:

- велика кількість програмних помилок протягом користування додатком;
- старий дизайн та інтерфейс, який не є user-friendly.

Цей програмний додаток має перевірену інформативну складову, але невдалу реалізацію. Це доводить велика кількість програмних помилок. Також, цей програмний додаток має низьку оцінку в App Store (2.4/5) та велику кількість негативних відгуків від реальних користувачів.

Також, а ході пошуку існуючих рішень було знайдено велику кількість програмних додатків, які направлені тільки на систему нагадувань, або тільки на відстеження певних показників організму. Але розроблювальна система повинна вирішувати всі поставлені проблеми.

Таблиця 1

Порівняльна характеристика програмних додатків

Назва додатка	Інтерфейс	Технології	Згода на обробку даних	Функції подій та нагадувань	Оцінка в App Store/ Play Market
«Моё здоровье: образ жизни»	Створено без вимог	хмарні технології, крос-платформна розробка	присутня (тільки для громадян РФ)	присутні	4.4/5
«Дневник здоровья»	Створено без вимог	нативна розробка (Android)	відсутня	відсутні	3.6/5
«Youwell»	Створено без вимог	нативна розробка (IOS)	відсутня	присутні	4.08/5

1.3. Аналіз існуючих технологій

1.3.1. *Огляд технології «Хмарне сховище»*

Хмарні технології – засоби АЗ та ПЗ, які реалізовують віддалений доступ та роботу з різними файлами для користувача. Користувач може працювати локально на своєму комп’ютері, використовуючи потужності АЗ. Завдяки хмарним технологіям, користувач може користуватись різними програмами та зберігати файли незалежно від характеристик свого пристрою. Єдиною умовою є доступ в мережу Інтернет [10]. Використання хмарних технологій позбавляє користувача використовувати різні накопичувачі, наприклад, жорсткі диски або флешки.

Переваги технології «Хмарне сховище»:

- надійність – зумовлюється тим, що використання повноцінних локальних ресурсів може бути проблематичним через вартість та складність підтримки;
- економічність – пропадає необхідність дорогого апаратного забезпечення для виконання комплексних задач. А також є можливість економити на роботі фахівця з налаштування та обслуговування цих пристроїв;
- мобільність – необхідність прив'язки до одного робочого місця пропадає через можливість мати доступ до необхідної інформації віддалено;
- доступність – користувач може доступитись до необхідної йому інформації через будь-який зручних для нього пристрій, підключений до мережі Інтернет.

Недоліки технології «Хмарне сховище»:

- обов'язкова необхідність доступу в Інтернет;
- при певних умовах використання хмарних сховищ, інформація може бути втрачена через несплату послуг;
- технологія не забезпечує абсолютну конфіденційність.

Приклади хмарних сервісів – Google Drive, Яндекс Диск.

1.3.2. *Огляд типів розробки мобільних додатків*

Нативний спосіб

При нативній розробці програмісти використовують притаманні мобільниій ОС мови програмування та інструменти мобільної ОС. Розробка iOS-додатка ведеться в інтегрованій ОС X та середовищі Xcode. Мови програмування: Objective-C, Swift, C і C ++. Для розробки ПЗ для Android використовуються середовище розробки Android Studio та мова програмування Java. Компанії Apple та Google надають всі необхідні

матеріали та удосконалюють інструменти розробки для поширення мобільних додатків.

Нативні додатки – це прикладні програми, які були розроблені для використання на визначених платформах або на визначених пристроях [11].

Переваги нативних програмних додатків:

- швидкість роботи мобільного додатка – досягається оптимізацією програмного коду під конкретні ОС;
- гнучкість реалізації – використовуються всі можливості мобільної ОС;
- простота тестування – всі параметри в процесі роботи програмного додатка контролюються автоматично;
- підтримка мобільного додатка від компанії-розробника.

Недоліки нативних програмних додатків:

- невиправдане використання в сферах ігрової розробки [12].

Крос-платформний спосіб

У крос-платформній розробці використовуються спеціальні інструменти (Unity, PhoneGap, Xamarin), які дозволяють створювати програмні додатки одразу для різних мобільних ОС. Це досягається за допомогою високорівневих мов програмування та інструментів, які надають можливість виконувати компіляцію одразу для декількох платформ.

Такий спосіб часто застосовують при розробці ігор, бо це значно зменшує дорогий період розробки та не шкодить якості отриманого продукту.

Переваги крос-платформної розробки:

- низька вартість – використання однієї технології і набору графіки знижує кількість робочих годин і бюджету проекту;
- тривалість розробки – відсутність унікальних елементів інтерфейсу і одна технологічна платформа скорочує терміни розробки;
- підтримка та оновлення продукту – додавання функціоналу або виправлення помилок відбувається одразу для всіх платформ;

– єдина логіка додатків – логіка додатків буде однаково працювати для всіх платформ. Написана і протестована логіка містить потенційно меншу кількість помилок у своїй роботі.

Недоліки крос-платформної розробки:

– повільна робота мобільного додатка;
– не використовуються унікальні особливості платформ;
– не інтуїтивно-зрозумілий інтерфейс для користувача.

1.4. Вимоги до програмного додатка контролю здоров'я

1.4.1. *Вимоги до інтерфейсу:*

– адаптованість – сумісність з можливостями та потребами користувача, простота переходу від виконання різних функцій, надання коректного зворотного зв'язку на запити користувача;
– доступність – однозначна реакція системи на різні типи запитів;
– юзабіліті – простота інтерфейсу, можливість задовольнити потреби користувача;
– гнучкість – адаптування інтерфейсу до вирішення певної задачі.

1.4.2. *Апаратні вимоги:*

– смартфон з ОС iOS (версія 8.0 і вище);
– доступ в мережу Інтернет (через Wi-Fi модуль або мобільний Інтернет) для зберігання даних на хмарному сервері.

1.4.3. *Вимоги до технологій:*

– використання технології «хмарного сховища» для можливості доступу інформації з віддаленого пристрою.
– Застосування нативного способу розробки для забезпечення швидкості коректної роботи мобільного додатка.

1.4.4. *Операційні вимоги*

- безпека та конфіденційність – запит на обробку персональних даних;
- надійність – шифрування паролів користувачів;
- відновлюваність – використання кешування даних;
- продуктивність – мінімальний час відклику при натисканні на екран у програмному додатка (0.3 секунди);
- збереження даних – використання бази даних для збереження даних користувача;
- керування помилками – коректне завершення роботи програмного додатка.

1.5. Висновки до розділу

В результаті аналізу цієї предметної області можна зробити висновок, що існує необхідність створення мобільного додатка для контролю здоров’я.

Можна сформулювати задачі, які повинен вирішувати мобільний додаток планування медичного обслуговування у лікарів первинної ланки медичної допомоги:

- план відвідування лікарів зі сталою періодичністю. Для цього необхідно реалізувати систему нагадувань відвідання лікаря та систему створення подій у календарі про відвідування лікаря;
- підвищити рівень загальних знань про контроль свого здоров’я, створити інформаційний розділ, де будуть опубліковані статті інформаційного характеру;
- план систематичного приймання медичних препаратів – створення функції створення власного графіка приймання ліків.

Були проаналізовані наявні рішення та технології, які використовуються для розробки мобільних додатків. Необхідно використання технології «хмарного сховища» для можливості доступу

інформації з віддаленого пристрою, та застосування нативного способу розробки для забезпечення швидкості коректної роботи мобільного додатка.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору ОС мобільного додатка

Розглянемо дві найбільш популярні мобільні ОС на сьогоднішній день та проаналізуємо, яку буде доцільно обрати.

2.1.1. *Огляд ОС iOS*

iOS – мобільна операційна система, яку створила та розробила компанія Apple. Вона призначена використовуватись тільки на апаратному забезпеченні компанії Apple. Це відмінна характеристика від мобільних ОС Android та Windows Phone. iOS – це друга за популярністю ОС у світі після Android (14% від кількості всіх проданих смартфонів за 2017 рік).

Для розповсюдження додатків компанія Apple використовує інтернет-магазин додатків App Store. За офіційною статистикою станом на березень 2018 року в App Store розміщено понад 2,1 мільйонів додатків (приблизно 1 млн створено для iPad). В порівнянні з початком роботи App Store у 2008 році, там було розміщено 500 програмних додатків [13].

Компанія Apple створила IDE Xcode для розробки під ОС iOS. IDE є зручною через редактор вихідного коду, де можна легко змінювати програмний код, швидко компілювати, запускати виконувані файли та тестові набори даних. Також Xcode надав можливість спільно працювати над програмним кодом у команді. Apple зробила цей процес безпечним та простим, використовуючи хмарні технології та унікальний ключ SSH. Ця можливість була надана за допомогою наступних продуктів від систем контролю версій:

– GitHub;

– Bitbucket;

– GitLab.com.

Також Xcode містить потужні засоби відлагодження, які значно економлять час розробника при пошуку програмної помилки. Якісний вбудований симулятор дозволяє легко тестувати розроблені мобільні

додатки, а зокрема інтерфейси. У розробника немає потреби володіти фізично всіма пристроями компанії Apple для успішного тестування. Всі необхідні симулятори моделей пристроїв та версій iOS містяться в IDE [14].

Перевагою мобільної ОС iOS є централізованість платформи. Всі пристрої одного користувача з'єднані одним акаунтом AppleID. Вся інформація буде синхронізуватись в одному місці, паролі зберігаються для всіх пристроїв і користувач відчуває себе в єдиній екосистемі, що безперечно є дуже зручним рішенням. Також, це полегшує роботу розробникам системи. Смартфони на базі iOS будуть підтримуватись компанією Apple, тож користувачі iOS-смартфонів можуть не турбуватись про помилки та неточності в роботі з системою, бо їх виправлять в наступній версії обновлень з великою ймовірністю [15].

Перевагою ОС iOS є те, що її користувачі є більш платоспроможними, тому що Apple завжди пропагувала якість та безпеку своїх програмних продуктів. Неможливо завантажити мобільний додаток на iPhone, без використання App Store. Всі сервіси iOS дуже добре захищені, бо компанія приділяє цьому питанню багато уваги. Це означає кращу перспективу мобільного додатка для монетизації [16].

Мови програмування для мобільної ОС iOS – Objective-C та Swift.

IDE – Xcode.

2.1.2. *Огляд ОС Android*

Android – мобільна операційна система з відкритим кодом, яку створила та розробила компанія Google. Основою для її створення було ядро Linux. Android – це найбільш популярна ОС у світі, яка використовується на смартфонах (починаючи з 2011 року). З цієї мобільної ОС частіше всього виходять в мережу Інтернет [17].

Для розповсюдження додатків для користувачів компанія Google використовує власний офіційний інтернет-магазин додатків Google Play. Також Google Play надає користувачам похідні сервіси для пошуку та

завантаження книг, музики та фільмів. За офіційною статистикою станом на кінець 2017 року в Google Play розміщено приблизно 3,5 мільйонів додатків. А за статистикою 2016 року програмні додатки з Google Play було завантажено 82 разів [18].

Для розробки додатків для ОС Android можна використовувати різні середовища розробки. Найбільш популярними рішеннями є Eclipse від компанії IBM та Android Studio від компанії Google. Перевагами Eclipse над іншими середовищами розробки є безкоштовне та безпечне використання. Завдяки таким сильним перевагам, компанія IBM заробила прихильність більшості Java розробників до сьогоднішнього дня. Eclipse містить велику кількість плагінів для виконання майже будь-якої задачі. Значним недоліком середовища розробки Eclipse є низька продуктивність через використання старої віртуальної машини JVM, яка була розроблена на початку 2000-х років разом з середовищем розробки.

Android Studio в порівнянні з Eclipse Android Studio має ряд переваг:

- вбудовані SDK, які автоматично завантажують необхідні пакети та інструменти для розробки;
- підтримка декількох мов програмування (Java, C, C++);
- підтримка компанії Google. Тобто через Android Studio компанія Google реалізовує своє бачення розвитку Android-додатків.

Але значним мінусом цієї IDE є повільна робота вбудованого емулятора Android. Для його коректної роботи розробник системи повинен мати якісне та потужне апаратне забезпечення, яке має високу вартість [19].

В порівнянні цих двох середовищ розробки з Xcode, то останній має всі переваги, що й IDE для Android-додатків, і не має їхніх недоліків.

Перевагою мобільної ОС Android є те, що користувачу доступно велика кількість пристроїв від різних виробників, з варіацією параметрів та ціновим діапазоном. Але для розробників це значний недолік. Через встановлення мобільної ОС Android на пристроях різних компаній, ОС втрачає свою якість, бо неможливо створити настільки універсальну ОС.

Для розв’язання цієї проблеми виробники використовують власні оболонки для коректної роботи пристрою з ПЗ від Google [20]. Найбільш відомі – Xiaomi, Meizu.

Через наявність великої кількості виробників Android-пристроїв, виникає проблема поєднання ПЗ та АЗ, так званого «заліза». Всі пристрої мають різну оптимізацію та витрачають ресурси по-різному. Навіть при більш продуктивному АЗ, пристрій на базі Android може працювати повільніше через недостатню оптимізацію ОС та поєднанню її з АЗ. Цієї проблеми немає в iOS-пристроїв через централізованість платформи. Тому доцільним є використання пристроїв на базі так званого «чистого» Android, розробкою яких займаються компанія Google [21]. Наприклад, лінія смартфонів та планшетів Nexus та Pixel.

Через ці особливості Android страждає інтерфейс ОС. Пристрої с ОС Android мають широку кількість розмірів екранів, тому розробники використовують XML файли для створення графічних інтерфейсів. Це означає, що вміст мобільного додатку відображається єдиним способом на екранах.

Мова програмування – Java.

Основні IDE – Eclipse, Android Studio.

Таким чином, проаналізувавши інформацію про мобільні ОС iOS та Android було виділено наступні переваги ОС iOS:

- гнучке середовище розробки Xcode, потужний інструмент тестування та симуляції;
- централізованість платформи, завдяки якій синхронізація у мобільному додатку може бути виконана набагато легше та безпечніше;
- висока платоспроможність користувачів iOS, що означає перспективу розвитку додатку в цілях монетизації.

Враховуючи переваги та недоліки кожної мобільної ОС, було обрано ОС iOS.

2.2. Обґрунтування вибору мови програмування

2.2.1. *Мова програмування Swift*

Swift – об’єктно-орієнтовна мова програмування, яку створила компанія Apple для розробки програмних додатків для ОС iOS та macOS. Основою для створення Swift стала мова програмування Objective-C. Swift замислювався як сучасна версія старої мови програмування Objective-C з простим синтаксисом, щоб зробити роботу розробника легше [22].

Платформа: iOS, macOS та інші платформи від Apple.

Основна IDE: Xcode.

Програмний код на Swift дуже легко читати, також вона стійка до помилок розробника в порівнянні з Objective-C. Як і багато високорівневих мов програмування Swift не дає доступу розробнику програмного додатка до внутрішніх файлів та низькорівневого коду, що зменшує ймовірність помилок від самого розробника. Swift продовжує деякі концепції Objective-C, наприклад, додавання методів до програмного коду під час його виконання. Також мова програмування Swift може запускатись в IDE Xcode за допомогою REPL. Тобто користувач IDE може одразу побачити результат свого програмного коду без повторного запуску системи. Водночас мова програмування Swift дуже швидка. Наприклад, сортування комплексних об’єктів виконується в 3,9 рази швидше, ніж в Python, і майже в 1,5 рази швидше, ніж в Objective-C.

Swift – це C-подібна мова програмування, але вона має деякі принципові відмінності від Objective-C. Наприклад, не обов’язково ставити крапку з комою після програмних виразів, немає заголовних файлів, повна підтримка Unicode, перевизначення операторів для класів, відбувається ініціалізація змінних та констант, а також меж масиву. Swift має строгу типізацію: в будь-який момент часу ви точно знаєте, з об'єктом якого типу ви працюєте.

Код, написаний на Swift, може працювати разом з кодом, написаним на мовах програмування C і Objective-C в рамках одного і того ж проекту, що означає гнучкість цієї мови програмування.

Переваги мови програмування Swift:

- інтуїтивна робота програм – програми починають працювати до того, як буде написаний весь програмний код;
- використання швидкого компілятора – в порівнянні з іншими інтерпретованими мовами PHP і JS. Зберігання програм у вигляді набору виконуваних файлів;
- гарантована безпека написаного програмного коду від розробників Apple;
- існування допоміжних інформативних ресурсів – товариство програмістів Swift та багато форумів;
- якісна навчальна література – офіційна документація від Apple, яка постійно оновлюється, безкоштовний посібник в iBooks, та популярна книга В. Усова «Swift. Основи розробки додатків»;
- зручна IDE Xcode – функція автоматичного доповнення (те, чого не вистачає популярному IDE Notepad.exe).

Недоліки мови програмування Swift:

- менш динамічний, ніж Objective-C;
- важче робота з C-кодом;
- мала кількість проектів написаних на Swift.

2.2.2. *Мова програмування Objective-C*

Objective-C – об’єктно-орієнтовна мова програмування, яку створила компанія Apple на початку 1980-х років для розробки програмних додатків для ОС iOS та macOS. Вона розроблена у вигляді набору розширень стандартної мови програмування C. Більшість програмних додатків у App Store написані саме мовою програмування Objective-C. До випуску

компанією Apple у 2014 році нової мови програмування Swift Objective-C була єдиною мовою програмування для створення iOS додатків [23].

Платформа: iOS, macOS та інші платформи від Apple.

Основна IDE: Xcode.

При створенні цієї мови програмування була вирішена проблема повторного використання програмного коду. Таке рішення підвищило продуктивність та якість написаного програмного коду. Objective-C слабо типізована мова програмування. Objective-C є розширенням мови програмування C, тому Objective-C може використовувати всі існуючі інструменти, бібліотеки для мови C. Objective-C підтримує перенавантаження оператора та забороняє множинне успадкування на відміну від C++.

Компанія Apple випустила Objective-C 2.0 у 2006 році, яка включала в себе функції актуальні для нового покоління розробки програмних додатків. Вони зробили синтаксис мови більш зрозумілим для пониження порогу входу для вивчення мови, покращили продуктивність, але оновлений garbage collector не був новим рішенням для поставленої проблеми сміття в програмному коді.

Переваги мови програмування Objective-C:

- більшість існуючого ПЗ для продукції компанії Apple написане на Objective-C (понад 1,2 мільйона програмних додатків);
- велика кількість технічної літератури та офіційна документація;
- динамічна типізація мови програмування;
- сумісництво з C-подібними мовами, що дозволяє використовувати компілятори та інструменти мови програмування C для розробки програмних додатків на Objective-C.

Недоліки мови програмування Objective-C:

- низька читабельність програмного коду;
- низька продуктивність мови через динамічну типізацію та повільна швидкість роботи розробників;

– різне розширення для файлів модулів та файлів заголовків, що є незручним для створення нових файлів;
– нерегулярні оновлення мови програмування;
– менша популярність в перспективі через Swift.

2.2.3. *Мова програмування Java*

Java – об’єктно-орієнтовна мова програмування, яку створила компанія Sun Microsystems (але пізніше її викупила Oracle) у 1995 році. За офіційною статистикою 2018 року, Java стала однією з найпопулярніших мов програмування згідно проаналізованому програмному коду на GitHub. Її використовують 9 мільйонів розробників.

Концепція мови програмування Java – мати якомога менше залежностей реалізації. Тобто програмний код, написаний та скомпільований на Java, може працювати без перекомпіляції на всіх платформах, в яких підтримується Java. Машинний код, в який компілюється програмний код на Java буде працювати на будь-якій JVM, не зважаючи на базову архітектуру комп’ютера. Java була створена на основі C та C++, про що стверджує схожий синтаксис, але Java є більш високорівневою мовою програмування [24].

Платформа: Android, Android Wear.

Основні IDE: Android Studio, Eclipse.

Java використовується як основа для Android SDK, а сама мобільна ОС Android використовує ядро Linux, який написаний на C. Машинний код Android SDK не сумісна з машинним кодом Java, тому Android SDK мусить працювати на власній віртуальній машині. Вона спеціально оптимізована під смартфони та планшети, де використовується мобільна ОС Android.

Переваги мови програмування Java:

– використання ООП та популяризація його серед широких мас розробників;
– простий синтаксис через вплив мов програмування C та C++;

– наявність функцій, що запобігають критичним помилкам, наприклад Security Manager;
– незалежність від платформи використання завдяки JVM;
– наявність багатопоточності;
– наявність АММ, що автоматично очищує пам’ять, якщо розробник не зробив цю дію вручну;
– підтримка великої кількості БД;
– наявність великої кількості навчальних матеріалів та офіційної технічної документації, що постійно оновлюється та доповнюється;
– оновлення старих версій мови від компанії Oracle.

Недоліки мови програмування Java:

– низька продуктивність через використання JVM та некоректну роботу АММ;
– відсутність зручних способів для створення графічного інтерфейсу користувача;
– платне використання Java SE 8 в комерційних цілях.

2.2.4. *Мова програмування Kotlin*

Kotlin – мова програмування зі статичною типізацією, яку створила компанія Jet Brains у 2011 році. Kotlin напряму взаємодіє з JVM. Концепція мови програмування Kotlin – бути більш лаконічною та безпечнішою, ніж Java та не такою складною як Scala. Компіляція програмного коду мовою програмування Kotlin є швидшою в порівнянні із Scala. 7 травня 2019 року відомий американський IT-ресурс Tech Crunch визнав мову програмування Kotlin кращою мовою Google для розробки програмних додатків для мобільної ОС Android [25].

Платформа: Android.

Основна IDE: Android Studio.

Використання мови програмування Kotlin для розробки Android-додатків виконується на основі Java 7 зі своїми надбудовами, такими як

обробка використання нульового вказівника та використання функцій розширення. В сумісництві з мовою програмування Java та IDE Android Studio мова програмування Kotlin надає поліпшену читабельність програмного коду, розширені можливості Android SDK і, як наслідок, прискорену розробку ПЗ [25].

Переваги мови програмування Kotlin:

- лаконічність мови. Kotlin потребує менше коду, ніж Java для однакового класу;
- зручні властивості класів, запобігання помилки ділення на нуль, функції розширення, зручне форматування рядків, ;
- підтримка мови програмування від Google, про що свідчать декілька стабільних версій;
- повна сумісність з Java. Класи, написані на Kotlin, можна використовувати в Java і навпаки в межах одного проекту;
- інтерфейси можуть реалізовувати методи.

Недоліки мови програмування Kotlin:

- невелика популярність мови програмування через використання Java;
- слабке просування для потенційних користувачів від компанії-розробника Jet Brains.

Таким чином, проаналізувавши інформацію про мови програмування для мобільних ОС iOS та Android було виділено наступні переваги мови програмування Swift над Objective-C:

- якісна та регулярна підтримка від компанії Apple. З початку випуску мови (з 2014 року) Apple було випущено 4 версії цієї мови. На сьогоднішній день остання версія 4.2. Це свідчить про те, що Swift активно розвивається для пропагується компанією Apple для використання. Сьогодні мова програмування Objective-C вже застаріла і майже не підтримується;

– швидка та зручна робота з програмним кодом Swift. Його набагато простіше читати розробнику системи, а також модифікувати та створювати. В Objective-C необхідно створити окремий файл для моделі та для хедера та логічно їх поєднати;
– офіційна технічна література від Apple та активне оточення розробників.

Враховуючи переваги та недоліки кожної мови програмування в цій роботі буде доцільно використовувати мову програмування Swift.

2.3. Обґрунтування вибору системи керування базами даних

2.3.1. *Firebase*

Firebase – платформа для розробки програмних додатків, яку розробила компанія Firebase. У 2014 році її викупила компанія Google. На даний момент Firebase має 18 програмних продуктів, які використовують 1,5 мільйони програмних додатків. Найвідоміші програмні додатки – аналітика Firebase, Firebase Cloud Messaging, Firebase Auth, Firebase Realtime, хостинг Firebase. Також, для мобільних ОС Android та iOS було створено інфраструктуру для тестування програмних додатків. Всі необхідні матеріали про результати тестування доступні в зручному вигляді. Є можливість автоматичного тестування при умові, що розробник ПЗ не передбачив тестовий код для розробленої системи [26].

Firebase Realtime – це сервіс, який надає БД, в основі якої лежить хмарна технологія. Дані зберігаються у форматі JSON і синхронізуються в реальному часі. Використовуючи SDK Firebase для розробки під мобільну ОС, то розробники отримують автоматичне оновлення з новими даними. Дані залишаються доступними при переході в автономний режим користування.

Основні функції Firebase:

– синхронізація даних в режимі реального часу;

– швидкий доступ до документів і колекцій, що зберігаються в БД;
– SDK для iOS, Android і Web з офлайн-доступом до даних;
– автоматична реплікація даних з сильною узгодженістю;
– серверні SDK для Node, Python, Go і Java;
– простота у використанні, що є пріоритетом для всіх продуктів Firebase;
– синхронізація даних між пристроями в режимі реального часу, що дозволяє створювати додатки з автоматичною синхронізацією даних;
– використання колекцій і документів для структурування та обробки даних. При цьому підході продуктивність запитів не залежить від обсягу даних, і результат роботи з базою з 100 і 100 мільйонів документів буде однаково ефективний;
– забезпечення безсерверної розробки. Клієнтський SDK Cloud Firestore сам піклується про забезпечення працездатності аутентифікації користувачів. Бекенд надає набір утиліт, які контролюють доступ до даних;
– повна інтеграція з платформою Firebase.

2.3.2. *MySQL*

MySQL – реляційна система управління реляційними базами даних, є СУБД з відкритим вихідним кодом. Була створена компанією «TcX» у 1995 році, нині нею володіє компанія Oracle. Метою створення MySQL було підвищити швидкість обробки великих об’ємів даних. Вона підтримується багатьма відомими мовами програмування. MySQL – це найбільш поширена серверна СУБД. MySQL часто використовується для веб-додатків, її використовують такі відомі сервіси як Facebook, Twitter, Flickr, YouTube.

Існує два видання для цієї СУБД: MySQL Server, який має відкритий вихідний код та фірмовий Enterprise Server. Останній відрізняється певною

кількістю власних розширень, які встановлюються як серверні плагіни, але обидва видання будуються на однаковій кодовій базі [27].

Основні функції MySQL:

– крос-платформне використання та підтримка;
– відповідність стандартам SQL;
– використання тригерів;
– використання курсорів для полегшення обробки запитів вилучення, додавання і видалення записів бази даних;
– наявність статистики про роботу сервера та продуктивність запитів;
– кількість рядків у таблицях до 50 мільйонів записів;
– дотримання властивостей ACID;
– кешування запитів;
– вкладені запити;
– вбудована підтримка реплікації;
– синхронна, асинхронна та напівсинхронна реплікація;
– підтримка Unicode;
– використання протоколу SSL для забезпечення додаткового захисту СКБД;
– повнотекстовий пошук та індексація;
– вбудована бібліотека баз даних;
– рідні системи зберігання: InnoDB, MyISAM, Merge, Memory, Federated, Archive, CSV, Blackhole, NDB Cluster.

Перевагами MySQL є простота в роботі, здатність до масштабування, швидкість роботи, надійність, якісне наслідування функціоналу від SQL, безкоштовне користування для некомерційних проектів.

Недоліками MySQL є менша надійність транзакцій через те, що при створенні СУБД їх не було розроблено, більш повільний процес розробки в порівнянні з сучасними БД, обмеження функціоналу, який не вистачає для розробки сучасних програмних додатків, невідповідність стандартам SQL в

певних принципових питаннях. MySQL не може виконувати багато процесів одночасно, що ставить її ефективність під сумнів.

Таким чином, проаналізувавши інформацію про сервіс Firebase Realtime та СУБД MySQL та порівнявши їх можливості, було виділено наступні переваги Firebase над MySQL:

- швидкодія роботи з даними. Розробник та користувач може дуже швидко доступитись до даних;
- простота в роботі з системою. Не потрібно створювати велику кількість файлів, треба лише зареєструвати програмний додаток на офіційному сайті Firebase;
- активна підтримка від Google та безкоштовний зворотній зв’язок від спеціалістів по розробці.

Враховуючи переваги та недоліки кожного способу зберігання даних в цій роботі буде доцільно використовувати сервіс для зберігання даних Firebase Realtime.

2.4. Шаблон проектування MVC

Model-View-Controller (MVC) – архітектурний шаблон проектування, відноситься до глобального шаблону проектування в проекті, який беруть за основу при проектуванні та розробці ПЗ. Суть цього шаблону проектування в поділі розроблюваної системи на три частини, які зв’язані між собою (див. рис. 2). Це модель даних, вигляд системи та система керування програмного додатка. Дуже часто цей шаблон проектування використовується для GUI у веб-додатках. Багато мов програмування, наприклад, JavaScript, Python, Ruby, PHP, Java, C# містять шаблони MVC для продуктивного створення програмних додатків.

Мета шаблону – створити гнучкий програмний додаток, в якому не буде виникати проблем при масштабуванні, зміні певного функціонала та дизайну. Також, другорядною метою є використання програмного коду декілька разів для більшої продуктивності програмного додатка. А також

відокремити логічне представлення програмного додатка від того, що бачить користувач. При використанні шаблону MVC великі програмні додатки стають більш зрозумілими для самих розробників та для користувачів як наслідок [28].

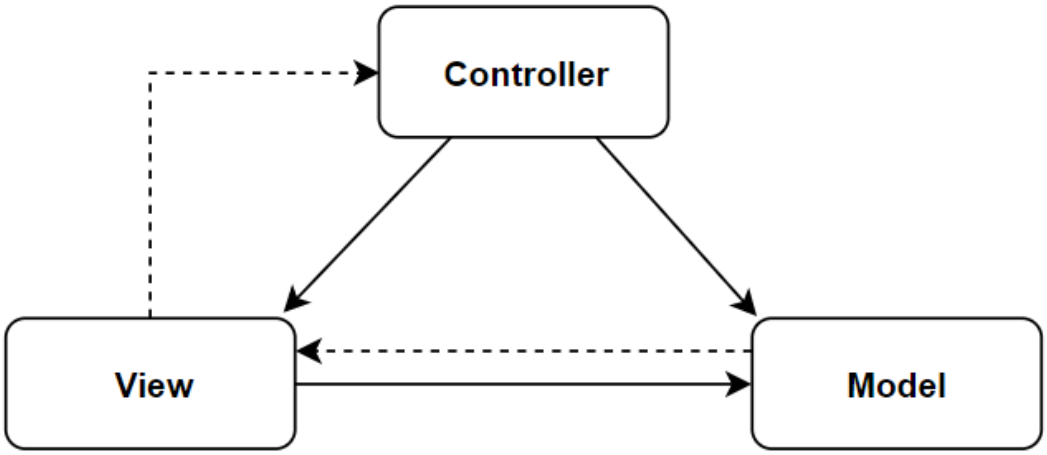


Рис. 2. Діаграма взаємодії між елементами шаблону MVC

Модель – одна с частин шаблону проектування MVC, яка відповідає за логіку програмного додатка. Модель повинна бути повністю незалежною від інших частин програмного додатка. Дані у моделі можуть змінюватись, але при цьому спосіб їх представлення повинен бути незалежним та незмінним.

Ознаки моделі:

– інкапсуляція ядра даних;

– містить логіку програмного додатка;

– як правило – це робота з БД.

Представлення – одна с частин шаблону проектування MVC, яка відповідає за представлення даних програмного додатка, які були отримані від моделі. Воно не може впливати на модель, а доступатись лише для отримання даних та реалізовувати їх відображення.

Приклади представлення – сторінка HTML, Windows Form.

Контроллер – одна с частин шаблону проектування MVC, яка відповідає за керування представленням програмного додатка. Контроллер

може вплинути на модель для надання представленню інших даних. Також, він відповідає за відстежування дій користувача – зміни положення курсора миші, натискання кнопки, ввід даних в текстове поле. Потім в залежності від типу дії, він передає запит до моделі або до представлення для виконання дії користувача.

Окрім MVC існують похідні шаблони проектування – MVP, MVVM.

Model-View-Presenter (MVP) – використовується, коли логічне поєднання даних неможливе. Прикладом використання цього шаблону є Windows Form.

Model-View-View Model (MVVM) – використовується, коли логічне поєднання даних можливе без використання інтерфейсів представлення. Прикладом MVP є технологія WPF.

У мові програмування Swift концепція шаблону проектування MVC досягається за рахунок стандартних інструментів розробки. В результаті використання MVC програмний додаток стає надійним та здатним до масштабування.

2.5. Висновки до розділу

В цьому розділі були розглянуті та проаналізовані основні інструменти для розробки мобільних додатків. Отримана інформація була систематизована з метою використання в подальшій роботі для визначення конкретних засобів реалізації, що будуть використовуватись при розробці автоматизованої системи планування медичного обслуговування.

Для розробки мобільного застосунку було вирішено обрати ОС iOS, тому що:

- більш зручні середовища розробки та інструмент для створення графічного інтерфейсу в порівнянні з Android;
- аудиторія користувачів iOS є більш платоспроможною;
- більш якісні програмні додатки через політику компанії Apple.

Для розробки мобільного застосунку з ОС iOS було обрано мову

програмування Swift, оскільки:

- зручність читання коду в порівнянні з Objective-C, де код є більш незрозумілим;
- швидкість створення та збірки проекту;
- зручна IDE XCode з функціями автодоповнення.

В якості системи керування базами даних вирішено обрати Firebase через її швидкодію. Доступ до цих файлів дуже простий як для користувача, так і для розробника. Нам не потрібно створювати велику кількість допоміжних файлів, натомість треба лише зареєструвати наш додаток на офіційному сайті Firebase.

Для побудови архітектури дипломного проекту був обраний шаблон MVC, оскільки він забезпечує легкість підтримки програми, головні компоненти якої можливо змінювати незалежно один від одного.

3. РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКА

3.1. Структура програмного додатка

Мобільний додаток з ОС iOS було розроблено з виконанням всіх вимог описаних в підрозділі 1.4.

Користувач мобільного додатка може зареєструватись у двох ролях: «Пацієнт» та «Лікар». Було вирішено розробити мобільний додаток для користувача зі статусом «Пацієнт», а в наступних версіях продукту додати функціонал для статусу «Лікар». При реєстрації користувач зі статусом «Пацієнт» (далі «користувач») вказує ім’я, дату народження, e-mail та пароль для подальшого внесення цієї інформації в особистий кабінет користувача. Для авторизації в системі користувач повинен ввести e-mail та пароль.

Зареєстрований та авторизований користувач отримує доступ до календаря, який є головною сторінкою мобільного додатка.

Також користувач має доступ до наступних сторінок:

- «Мій акаунт». Можна переглядати та редагувати дані у своєму особистому кабінеті;
- «Календар». Можна додавати та видаляти замітки, графік ліків та нагадування про прийом до лікаря;
- «Тематичні статі». Можливість читати тематичні статі, перевірені медичними експертами;
- «Лікарі». Можливість переглядати лікарів та інформацію про них;
- «Ліки». Можливість додавати ліки та формувати правила їх прийому;
- «Історія захворювань». Можливість переглядати історію захворювань та зберігати її у форматі .pdf.

Також у меню мобільного додатка є список сторінок для просування та покращення розробленого додатка – «Запросити друзів», «Оцінити нас», «Зворотній зв’язок», «Про додаток».

3.2. Аналіз функціональних вимог до програмного додатка

Таблиця 2

Функціональні вимоги до автоматизованої системи планування медичного обслуговування у лікарів первинної ланки медичної допомоги

Код вимоги	Назва вимоги	Пріоритет вимоги	Коментар
01	Реєстрація в системі	1	З обов’язковим вказанням e-mail та паролю
02	Авторизація в системі за допомогою e-mail та паролю	1	
03	Створення графіку прийому ліків	1	За правилами періодичності
04	Додавання нагадування прийому ліків	2	
05	Додавання нагадування прийому у лікаря	2	
06	Додавання текстових заміток у календарі	3	
07	Редагування даних в особистому кабінеті	3	
08	Перегляд тематичних медичних статей	2	Всі статті мають рекомендаційний характер
09	Перегляд списку лікарів та інформації про них	1	Спеціалізація, стаж, номер телефону, клініка
10	Додавання історії захворювання та супутньої інформації	1	Назва, дати, опис
11	Зберігання історії захворювань в форматі .pdf	1	

Продовження табл. 2

12	Можливість залишити відгук	4	На AppStore
13	Можливість розповісти друзям у соціальних мережах	4	Telegram Facebook Instagram
14	Можливість написати листа розробникам	4	З метою удосконалення додатка
15	Можливість прочитати інформацію про розроблену систему	4	

3.3. Архітектура програмного додатка

3.3.1. *Опис системних файлів та бібліотек для БД*

В теці Other files зберігаються системні файли, які містять свої особливості:

- Info.plist (information property list file) – файл з системними налаштуваннями, в якому розробник вказує назву проекту та інші параметри (наприклад, дозвіл на запит про увімкнення мікрофону на клавіатурі) [29];
- AppDelegate.swift – клас з налаштуваннями, які необхідні мобільному додатку при його запуску. В цьому класі підключається Firebase та вказується шлях до БД;
- Main.storyboard – механізм розробки інтерфейсу мобільного додатка, в якому розробник вказує зв’язок між екранами, розміщує кнопки, тобто створює взаємодію між всіма графічними елементами мобільного додатка [30];
- Assets.xcassets (asset catalogs) – каталог, в якому завантажуються всі зображення, які будуть використані у мобільному додатку [31];

– LaunchScreen.storyboard – екран запуску, який з’являється одразу після запуску програми. Цей екран швидко змінюється на перший екран розробленого програмного додатка [32];
– GoogleService-Info.plist – файл з налаштуваннями Firebase (id, api key та інші поля).

Використана у розробленому мобільному додатку БД Firebase має наступні бібліотеки:

– Firebase Core – використовується для синхронізації БД та проекту;
– Firebase Auth – використовується для авторизації та реєстрації та для зберігання користувачів у системі. Також за допомогою можливостей цієї бібліотеки користувач може увійти до мобільного додатка з будь-якого пристрою. Ця бібліотека дає можливість виконувати авторизацію за допомогою різних способів: e-mail, e-mail та пароль, номер телефону, Google акаунт, Facebook, Twitter, Github, Microsoft акаунт, анонімний вхід. Проаналізувавши всі способи, було обрано спосіб за допомогою e-mail та паролю, тому що це найбільш надійний спосіб. У майбутньому буде додано способи за допомогою Google акаунту та Facebook;
– Firebase Database – бібліотека для запису даних в БД. Firebase використовує синхронізацію даних (а не HTTP-запити), тобто завжди при зміні даних, кожен підключений до сервера пристрій отримує оновлені дані дуже швидко. Також, через чуйність в автономному режимі, мобільні додатки, що підключені до Firebase отримують зміни через синхронізацію мобільного додатка та поточного стану сервера. Також, через використання хмарних технологій, всі дані будуть доступні користувачеві з будь-якого пристрою та це використовує менше ресурсів.
– Firebase Storage – це бібліотека для завантаження, зберігання та вивантаження великих об’єктів до Google Cloud Storage. Вона

необхідна для реалізації вимоги зберігання фалів у форматі .pdf та для роботи з картинками в мобільному додатку;

– WYPopoverController – бібліотека для реалізації випадаючого списку [33]. В ОС iOS немає готової реалізації випадаючого списку, і одне з рішень – це використання сторонньої бібліотеки.

3.4. Реалізація шаблону проектування MVC

3.4.1. *Models*

В розробленому мобільному додатку існують наступні моделі: CalendarNote, DoctoreNote, Doctor, HistoryElement, Specialization, PillElement та PillCategory. Ці моделі були створені для реалізації вимог до мобільного додатка та потрібні для коректного запису у БД.

– CalendarNote – це структура, в якій представлена модель заміток, які користувач додає у календарі. Вона складається з наступних полів – місяць замітки, день замітки, текстове поле замітки, спеціалізація лікаря, ім’я лікаря, замітка про лікаря, час лікаря, назва таблетки, замітка таблетки. Значення всіх полів за замовчуванням дорівнюють null. Ця структура прив’язується до кожної обраної дати на календарі. Замітка створюється, якщо ми маємо хоч одне заповнене поле в структурі «Замітка», «Лікар» або «Ліки». Якщо всі поля пусті, то замітка не створюється;

– DoctoreNote – структура замітки про лікаря, яка використовується при натисканні на кнопку «Прийом до лікаря» при виборі дати в календарі. Вона складається з наступних полів – спеціалізація лікаря, ім’я лікаря, текстова замітка. Ці поля стосуються саме лікаря. Значення всіх полів за замовчуванням дорівнюють null. Якщо всі поля пусті, то замітка не створюється;

– Doctor – структура лікаря, яка відображає інформацію про лікаря. Вона складається з наступних полів – спеціалізація лікаря, ім’я

лікаря. Ці поля стосуються саме лікаря. Значення всіх полів за замовчуванням дорівнюють null. Якщо всі поля пусті, то лікар не може додатись у БД та у систему;

– HistoryElement – структура, в якій представлена хвороба зі списку хвороб. Вона складається з наступних полів – назва хвороби, діапазон дат хвороби (тривалість) та опис хвороби. Значення всіх полів за замовчуванням дорівнюють null. Якщо всі поля пусті, то хвороба не буде створена;
– Specialization – структура для опису лікарів кожної спеціалізації. Вона складається з наступних полів – назва спеціалізації та масив всіх лікарів, які мають таку ж спеціалізацію. Значення всіх полів за замовчуванням дорівнюють null;
– PillElement – це структура, в якій представлена модель ліків, які користувач додає у меню з ліками. Вона складається з наступних полів – категорія ліків, назва ліків, частота прийому ліків, тривалість прийому ліків, опис ліків. Значення всіх полів за замовчуванням дорівнюють null;
– PillCategory – структура категорії ліків (в розробленому програмному додатку маємо наступні категорії ліків – таблетка, мазь, краплі, інше). Складається з поля – назва ліків.

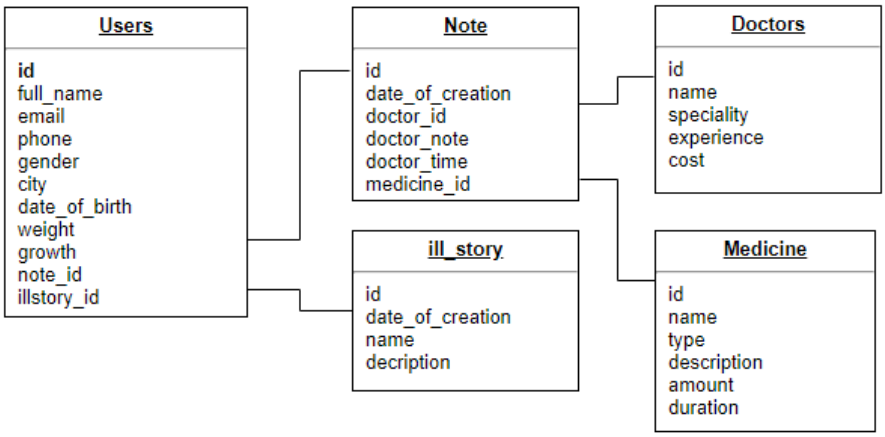


Рис. 3. Схема представлення БД

3.4.2. *Controllers*

- ContainerVC – клас-контейнер, в якому знаходяться два елементи мобільного додатка – календар та головне меню. Календар – це основний елемент цього екрану, але меню накладається зверху календаря при певній дії. Реалізована дія – висувати меню зліва направо поверх календаря. Під час знаходження меню на головній сцені дати на календарі стають неактивні. Користувач може вернути меню назад і натиснути на дату календаря;
- SideMenuVC – клас, який відповідає за роботу меню мобільного додатка. В цьому класі реалізовано логічні переходи на кожну сторінку з пунктів висувного меню. Даний клас відображається у вигляді таблиці;
- SliderVC – клас, який наслідується від стандартного класу ScrollView. ScrollView – це вид, який відповідає за рух елементів на екрані вліво та вправо. Даний клас відповідає за всі слайдери в мобільному додатку (вхід та реєстрація, зміна картинок в розділі зі статтями);
- ViewController – клас, який відповідає за головний екран календаря, а також за кнопки зміни теми мобільного додатка (темна або світла теми). В цьому класі реалізовано взаємодія кнопок, які з’являються при подвійному натисканні на дату в календарі («Прийом до лікаря», «Ліки», «Замітка»);
- NoteVC – клас, в якому реалізовано роботу на екрані запису замітки на певну дату в календарі. Замітка – це текстове поле, яке необхідно записати в БД. Після цього значення замітки змінюється, а також змінюється колір ланки календаря (це означає, що в цю дату вже створена замітка). Якщо користувач збереже пусте поле, то запис з БД видалиться;
- RegistrationVC – клас, в якому реалізований екран реєстрації користувачів. У цьому класі реалізовано вибір ролі користувача

системи «Пацієнт» або «Лікар». Якщо обрана роль користувача «Пацієнт», то наявні поля на екрані – ім’я користувача, дата народження, місто, e-mail, пароль, повторний пароль та checkbox «Прийняти правила використання мобільного додатка». Якщо обрана роль користувача «Лікар», то наявні поля на екрані – ім’я користувача, спеціалізація лікаря, лікарня, в якій працює лікар, e-mail, пароль, повторний пароль та checkbox «Прийняти правила використання мобільного додатка». Кнопка «Зареєструватись» стане доступною тільки у разі заповнення всіх полів, а також коли співпадають паролі та натиснутий checkbox щодо правил використання мобільного додатка. Якщо все дані пройшли валідацію, то в БД зберігається запис про цього користувача;

- AuthorizationVC – клас, в якому реалізована взаємодія на екрані авторизації. Також користувач обирає роль, під якою він хоче авторизуватись у мобільному додатка. Поля, які необхідно заповнити – e-mail та пароль. Якщо введені дані співпадають з даними, які зберігаються в БД, то користувач стає авторизованим у системі. Якщо дані не співпадають, то на екран виводиться повідомлення про те, що дані невірні;
- HistoryVC – клас, в якому реалізована взаємодія на екрані з історіями хвороби. Клас реалізований за допомогою таблиці, в яку додані всі хвороби. Якщо натиснути на строку з хворобою, то користувач може переглянути інформацію про хворобу. Натиснувши на кнопку зі знаком плюса, можна додати нову хворобу та заповнити інформацію про неї. Видалити хвороби зі списку можна пересунувши строку вліво. Всі дані зберігаються в БД та автоматично оновлюються;
- NewElementHistoryVC – клас, в якому реалізована взаємодія на екрані з додаванням нової хвороби. Клас NewElementHistoryVC має три поля – назва хвороби, період хвороби, інформація про хворобу;

– ChangeHistoryVC – клас, який відповідає за реалізацію зміни інформації про створену хворобу. Щоб змінити інформацію про створену хворобу, користувач повинен натиснути на строку хвороби у списку хвороб;
– DoctorNoteVC – клас, в якому реалізована взаємодія на екрані замітки до лікаря на календарі. Цей екран містить два поля з випадаючими списками – спеціалізація лікаря та ім’я лікаря. Також містить поля – замітка про прийом до лікаря та час прийому до лікаря. Якщо всі дані введено правильно, то при натисканні кнопки «Зберігти», дані вносяться в БД і ця дата на календарі змінює колір, що служить індикатором того, що в цей день створена замітка;
– LimitListVC – клас, в якому реалізовані всі випадаючі списки, які використовуються у додатка. Тут доцільно використовувати бібліотеку WYPopoverController, яку було описано в пункті 3.3.1;
– PillsVC – клас, в якому реалізована взаємодія на екрані ліків. Клас реалізований за допомогою таблиці, в яку додані всі ліки. Якщо натиснути на строку з ліками, то користувач може переглянути інформацію про них. Натиснувши на кнопку зі знаком плюса, можна додати нові ліки та заповнити інформацію про них. Видалити ліки зі списку можна пересунувши строку вліво. Всі дані зберігаються в БД та автоматично оновлюються;
– NewPillVC – клас, в якому реалізована взаємодія на екрані з додаванням нових ліків. Клас NewPillVC має три поля – назва ліків, кількість разів прийому на день, інформація про ліки в текстовому форматі;
– ChangePullVC – клас, який відповідає за модифікацію інформації про існуючі ліки.;
– PillNoteVC – клас, в якому реалізована взаємодія на екрані замітки до ліків на календарі. Цей екран містить поле з випадаючим списком всіх ліків, які користувач створював до цього. Також містить поле –

замітка про прийом ліків. Якщо всі дані введено правильно, то при натисканні кнопки «Зберігти», дані вносяться в БД і ця дата та всі наступні згідно періодичності прийняття ліків на календарі змінює колір, що служить індикатором того, що в цей день створена замітка.

3.4.3. *Views*

- Slide.xib – вид за замовчуванням для першого слайдера в мобільному додатку. В цьому файлі обрано колір слайдера та текст, який буде з’являтись на різних скріншотах слайдера;
- SlideView – клас, в якому реалізований вид для першого слайдера;
- WeekdaysView – клас, в якому реалізований вид календаря, а саме його верхньої частини, яка відображає дні тижня ("Su", "Mo", "Tu", "We", "Th", "Fr", "Sa");
- MonthView – клас, в якому реалізований вид календаря, що відповідає за відображення місяцю та року. Також в цьому класі реалізована логіка кнопок, які відповідають за перехід на наступний місяць або на попередній;
- CalendarView – клас, в якому реалізований вид самого календаря. А саме реалізована логіка днів та місяців, передбачені винятки, наприклад, «29 лютого». А також реалізовані натискання (одне чи два) на певну дату, зміна кольорів в залежності від натискання, кількість строк в календарі, висота кожної ланки, позиціонування ланки відносно інших (відстань між ланками). Передбачені випадки переключання з однієї ланки на іншу, та реалізована взаємодія з іншими видами, які наявні на цьому екрані (замітка, запис до лікаря, запис ліків);

3.5. Висновки до розділу

В цьому розділі було представлено загальний опис розробленої автоматизованої системи планування медичного обслуговування у лікарів первинної ланки медичної допомоги.

Були представлені функціональні вимоги мобільного додатка. Всі вимоги були реалізовані у системі.

Також були розглянуті використані технології, зокрема використання архітектурного шаблону Model-View-Controller, що є одним з найпопулярніших, найбільш використовуваних та ефективних шаблонів для розроблення сучасних мобільних додатків. У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов’язані частини з розподілом функцій між компонентами.

Для реалізації шаблону Model-View-Controller був використаний IDE XCode 10, що значно полегшує написання коду на Swift 4.2 для ОС iOS.

4. АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1. Тестування програмного додатка

Тестування програмного додатка – це один з важливих кроків розробки ПЗ. Метою тестування програмного додатка є отримання інформації щодо якості програмного продукту [34]. Також тестування ПЗ запобігає низці ризиків, що виникають при використанні програмного додатка користувачем.

Розробка цього програмного додатка відбувалась за допомогою методології SCRUM, тому проміжне тестування ПЗ 3 рази після кожного спринта. Для проміжного тестування було обрано ручне тестування, QA-спеціалістом якого був розробник програмного додатка. Під час проміжного ручного тестування було знайдено 20% помилок всього проекту. Отже, проміжне тестування під час розробки значно скорочує час загального тестування та виправлення помилок.

Тестування розробленого програмного додатка відбувалось у декілька етапів. Спочатку були створені автоматичні тести для тестування основних функцій програмного додатка для перевірки коректності роботи розроблених класів та взаємодії одне з одним. Для того, щоб провести такого роду тестування було проведено реорганізацію ділянок коду, які будуть тестуватись. Реорганізація проходила за наступними правилами:

- кожен окремий фрагмент коду повинен бути окремим об’єктом і він не має доступу до інших об’єктів. Це допомагає уникнути заплутаного коду;
- підтримування можливості налаштування замість того, щоб ставити конкретні значення певним параметрам;
- кожен метод будь-якого класу повинен бути простим і коротким для інтуїтивного розуміння фрагменту коду;

– для створення об’єктів класів слід використовувати конструктори. Це допоможе створенню правильних копій об’єкту класу для коректного тестування.

Автоматичне тестування відбувалось за допомогою технології Appium. Це інструмент для автоматизованого тестування нативних мобільних додатках з ОС iOS. Appium є крос-платформним, це означає, що: він дозволяє створювати автоматизовані тести на декількох платформах використовуючи один і той самий API. Розробник має змогу використовувати код повторно з тестовими наборами для iOS, Android і Windows.

Після успішного проведення автоматичного тестування було застосовано тестування методом «чорного ящика». Була створена фокус-група – 20 зацікавлених осіб, які виявили бажання протестувати цей продукт. Кожен з двадцяти людей користувались програмним додатком протягом десяти хвилин. При переході з одного екрану на інший вони заповнювали попередньо сформований лист, де треба було вказувати проблеми та враження від користування програмним продуктом. Також, їм необхідно було записати те, що вони хотіли б змінити в додатку та які функції додати. Цей лист був сформований згідно вимог розроблених у розділі 3 (підрозділ 3.2).

У процесі цього типу тестування було виявлено кілька незначних помилок у створені та взаємодії інтерфейсу:

– було знайдено граматичну помилку у тексті повідомлень;
– затримка у 2 секунди при натисканні на пункт в меню «Тематичні статі».

Перша помилка була зроблена через людський фактор, яку неможливо було відслідкувати під час автоматичного тестування. Друга помилка була пов’язана з особливістю реалізації цієї функції і після коректування реалізації ця проблема зникла.

Після розробки програмного додатка було виділено 30 робочих годин на тестування продукту, а також 60 робочих годин на усунення помилок, які можуть виникнути при фінальному тестуванні. Отже, розробник ПЗ мав достатню кількість часу для своєчасного завершення програмного додатка.

4.2. Порівняння програмного додатка з аналогами

В першому розділі дипломного проекту були проаналізовані три аналоги розробленого програмного додатка та представлені у вигляді порівняльної характеристики (див. табл. 1). Розробка цього програмного додатка відбувалась згідно розроблених вимог до ПЗ та з урахуванням переваг та недоліків існуючих аналогів.

Отриманий після розробки мобільний додаток відповідає всім розробленим вимогам до ПЗ та перевагам існуючим рішенням:

- можливість мати декілька акаунтів в одному додатка;
- містить функції відстеження показників здоров'я (ріст, вага, тиск);
- використання надійного криптографічного протоколу TLS/SSL при передачі даних;
- реєстрація за допомогою e-mail та паролю (зв'язок з поштовою скринькою).

Окрім реалізації переваг існуючих програмних продуктів, були усунуті наступні виявлені недоліки аналогічних рішень при розробці мобільного додатка:

- інтерфейс додатка створено з урахуванням вимог до проектування інтерфейсу;
- є можливість історії захворювань в форматі .pdf.
- програмний додаток функціонує в межах України, отже приймаючи користувацьку згоду, громадянин України зможе відстояти свої права в разі судових справ;

– наявність функцій створення подій (прийом ліків, відвідування лікаря) та нагадувань про подію, що є важливими функціями у вирішенні поставленої задачі;
– програмні помилки відсутні.

Також, в розробленому програмному продукті наявні функції для подальшого покращення мобільного додатка (залишити відгук, запросити друзів, написати листа розробникам з пропозиціями покращення), що стверджує про те, що цей програмний продукт створений с перспективою подальшого поліпшення.

Проаналізувавши отриманий програмний додаток, можна стверджувати, що розроблений продукт відповідає всім потребам користувача та є конкурентом вже існуючим рішенням.

4.3. Пропозиції для майбутнього поліпшення системи

Розроблений веб-додаток є мінімально-працюючим продуктом, що задовольняє критерії бакалаврського дипломного проекту, але для виходу на повноцінний ринок з можливістю подальшої монетизації розроблених функцій недостатньо. Запити користувачів швидко зростають, а конкуренти працюють над покращенням своїх продуктів, тому розроблений програмний додаток можна вдосконалити в наступних версіях продукту.

Також, при тестуванні системи фокус-групою було отримано ряд рекомендацій для покращення системи, які можна додати в наступні версії:

– реалізація функціоналу для ролі користувача «Лікар» та взаємодію лікарів та пацієнтів;
– можливість онлайн зв’язку лікарів та пацієнтів через чат або відео-зв’язок;
– можливість реєструватись та авторизуватись за допомогою Google акаунту, Facebook акаунту;
– додання можливості пошуку лікарів по спеціалізації;
– додання можливості сортування лікарів по стажу, районам

– можливість залишати відгук про лікаря у розробленій системі;
– можливість записатись автоматично на прийом до лікаря та налаштувати синхронізацію онлайн-запису та нагадувань про запис;
– можливість робити покупки в мобільному додатку (оплачувати онлайн-консультацію лікаря);
– додавання нової сторінки у меню, що відповідає за статистку показників людини;
– створення накопичувальної системи в мобільному додатку, створення акцій та знижок для користувачів з певним статусом.

Також, в розробленому програмному продукті наявні функції для подальшого покращення мобільного додатка (залишити відгук, запросити друзів, написати листа розробникам з пропозиціями покращення), що стверджує про те, що цей програмний продукт створений с перспективою подальшого поліпшення.

4.4. Висновки до розділу

В цьому розділі було розглянуто тестування розробленої автоматизованої системи планування медичного обслуговування у лікарів первинної ланки медичної допомоги.

Були розглянуті методи тестування розробленого програмного додатка. Тестування відбувалось в декілька етапів. Спочатку були створені автоматичні тести для тестування основних функцій програмного додатка для перевірки коректності роботи розроблених класів та взаємодії одне з одним. Автоматичне тестування відбувалось за допомогою технології Appium. Після успішного проведення автоматичного тестування було застосовано тестування методом «чорного ящика». Була створена фокус-група – 20 зацікавлених осіб, які виявили бажання протестувати цей продукт. Всі помилки у роботі програми, які були знайдені, були усунуті.

Був проведений аналіз розробленого програмного додатка з аналогами. Отриманий програмний продукт реалізовує всі переваги

існуючих програмних рішень, а також покриває своїм функціоналом певні недоліки.

Були виокремлені пропозиції для майбутнього покращення системи та проаналізовані наявні функції, що забезпечують отримання зворотного зв’язку від користувачів програмного додатка.

ВИСНОВКИ

В результаті розробки було створено програмний мобільний додаток для планування медичного обслуговування лікарів первинної ланки медичної допомоги. Мобільний додаток створено для користувачів – людей, які відвідують лікарів для планових або термінових оглядів, а також мають потребу в системі нагадувань та планування медичного обслуговування.

Було досліджено актуальність обраної тематики шляхом аналізування статистичних даних за останні п’ять років, а також пошуку прогнозів на майбутні десять років.

Був проведений аналіз існуючих програмних рішень шляхом порівняння переваг та недоліків аналогічних мобільних додатків для різних платформ та створено таблицю порівняння. Після цього було зроблено висновок щодо створення програмного рішення у вигляді мобільного додатка для смартфону та обраний спосіб розробки мобільного додатка. Також, були визначені функціональні вимоги до розроблюваної системи, серед яких є апаратні вимоги, вимоги до інтерфейсу, вимоги до технологій.

Після аналізу можливих засобів реалізації було прийняте рішення розроблювати мобільний додаток на основі ОС iOS на мові програмування Swift з середовищем розробки Xcode, використовуючи БД Firebase, і розробляти мобільний додаток згідно принципів ООП, а саме використовувати архітектурний шаблон MVC, який є найбільш популярним серед існуючих шаблонів проектування.

Розроблений програмний додаток має наступні функції:

- реєстрація користувача за допомогою e-mail та паролю;
- авторизація користувача;
- роботу з календарем, на якому можна додавати графік прийому ліків, подію про відвідування лікаря та текстову замітку;
- перегляд інформаційні тематичні статі рекомендаційного характеру;

– додавання нових ліків в профілі з правилами прийому та необхідною інформацією для створення графіку прийому ліків;
– додавання історії захворювання
– можливість залишити відгук або побажання.

Мобільний додаток розроблений в повному обсязі з відповідністю до розроблених вимог. Тестування мобільного додатка виконано в повному обсязі згідно наявної методики тестування.

Розроблений мобільний додаток допомагає користувачам економити свій час та створити систему власного медичного обслуговування у своєму смартфоні.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. List of countries by number of Internet users: [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/List_of_countries_by_number_of_Internet_users.– (23.05.2019).
2. Исследование: 45% всех взрослых жителей Украины уже пользуются смартфонами, а к 2020 году их проникновение возрастет до 70% [инфографика]: [Електронний ресурс]. – Режим доступу до ресурсу: https://itc.ua/news/issledovanie-45-vseh-vzroslyih-zhiteley-ukrainyi-uzhe-polzuyutsya-smartfonami-a-k-2020-godu-ih-proniknovenie-vozrastet-do-70-infografika. – (23.05.2019).
3. Разработка и отладка приложений для Andriod Wear: [Електронний ресурс]. – Режим доступу до ресурсу: https://software.intel.com/ru-ru/android/articles/android-wear-through-adb. – (15.03.2019).
4. Проблеми контролю здоров’я (анкета для споживачів): [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://docs.google.com/document/d/1_y7G4rOhc0-JdWFs3r2MvCaNikwxG0CZWnV5J1tQZNQ. – (15.03.2019).
5. Полное обследование организма, или что такое чекап и зачем он нужен: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.kp.ru/guide/polnoe-obsledovanie-organizma.html. – (10.05.2019).
6. «Моё Здоровье – образ жизни»: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://play.google.com/store/apps.medsolutions.myhealth. – (20.03.2019).
7. Моё Здоровье: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://myhealthapp.ru. – (20.03.2019).

8. Дневник здоровья: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=ru.shokurov.unitable4. – (20.03.2019).
9. Youwell: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://itunes.apple.com/ua/app/youwell. – (20.03.2019).
10. Облачные технологии: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: http://www.wikireality.ru/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8. – (20.03.2019).
11. Нативные приложения: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5. – (20.03.2019).
12. Нативная или кроссплатформенная разработка – что лучше?: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://wnfx.ru/nativnaya-ili-krossplatformennaya-razrabotka-chto-luchshe/. – (20.03.2019).
13. iOS: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/IOS. – (10.04.2019).
14. Xcode 10: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://developer.apple.com/xcode/. – (10.04.2019).
15. Война миров: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://rg.ru/2017/10/25/chem-ios-luchshe-android-i-chem-huzhe.html. – (10.04.2019).
16. iOS или Android: выбираем платформу для мобильного приложения: [Електронний ресурс]. – 2017. – Режим доступу до

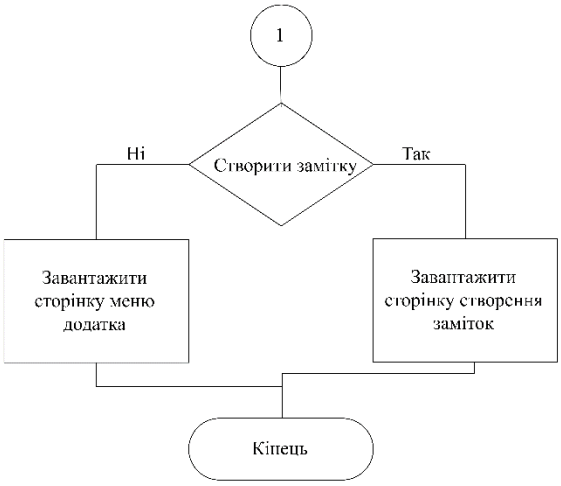
ресурсу: https://punicapp.com/blog/pages/544/ios-ili-android-vybiraem-platformu-dlya-mobilnogo-prilozheniya. – (10.04.2019).
17. Android (operating system): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Android_(operating_system). – (10.04.2019).
18. Google Play: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Play. – (10.04.2019).
19. Android Studio: среда разработки мобильных приложений: [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://arduinoplus.ru/android-studio. – (10.04.2019).
20. Android vs iOS: в чем разница?: [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://observer.com.ua/android-vs-ios-v-chem-raznitsa. – (10.04.2019).
21. iOS против Android. Что всё-таки лучше?: [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.iphones.ru/iNotes/759337. – (10.04.2019).
22. Swift (programming language): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Swift_(programming_language). – (20.04.2019).
23. Objective-C: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Objective-C. – (20.04.2019).
24. Java (programming language): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Java_(programming_language). – (20.04.2019).
25. Kotlin (programming language): [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:

https://en.wikipedia.org/wiki/Kotlin_(programming_language). – (20.04.2019).
26. Firebase: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Firebase. – (20.04.2019).
27. MySQL: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MySQL. – (20.04.2019).
28. Model–view–controller: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. – (20.05.2019).
29. About Information Property List Files: [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/AboutInformationPropertyListFiles.html. – (2.05.2019).
30. Использование Storyboard: [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://habr.com/ru/post/131128. – (2.05.2019).
31. Adding Images: [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/AddingImages.html. – (2.05.2019).
32. Launch Screen: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/launch-screen. – (2.05.2019).
33. WYPopoverController: [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://github.com/nicolaschengdev/WYPopoverController. – (2.05.2019).
34. Software testing: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Software_testing. – (2.05.2019).

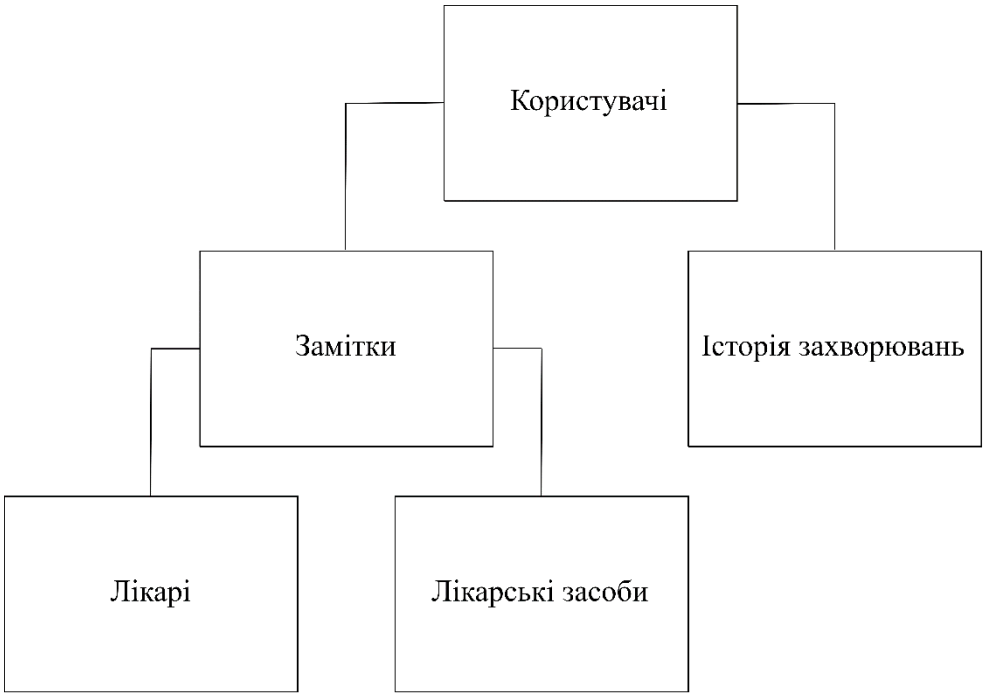
ДОДАТКИ

Додаток 1

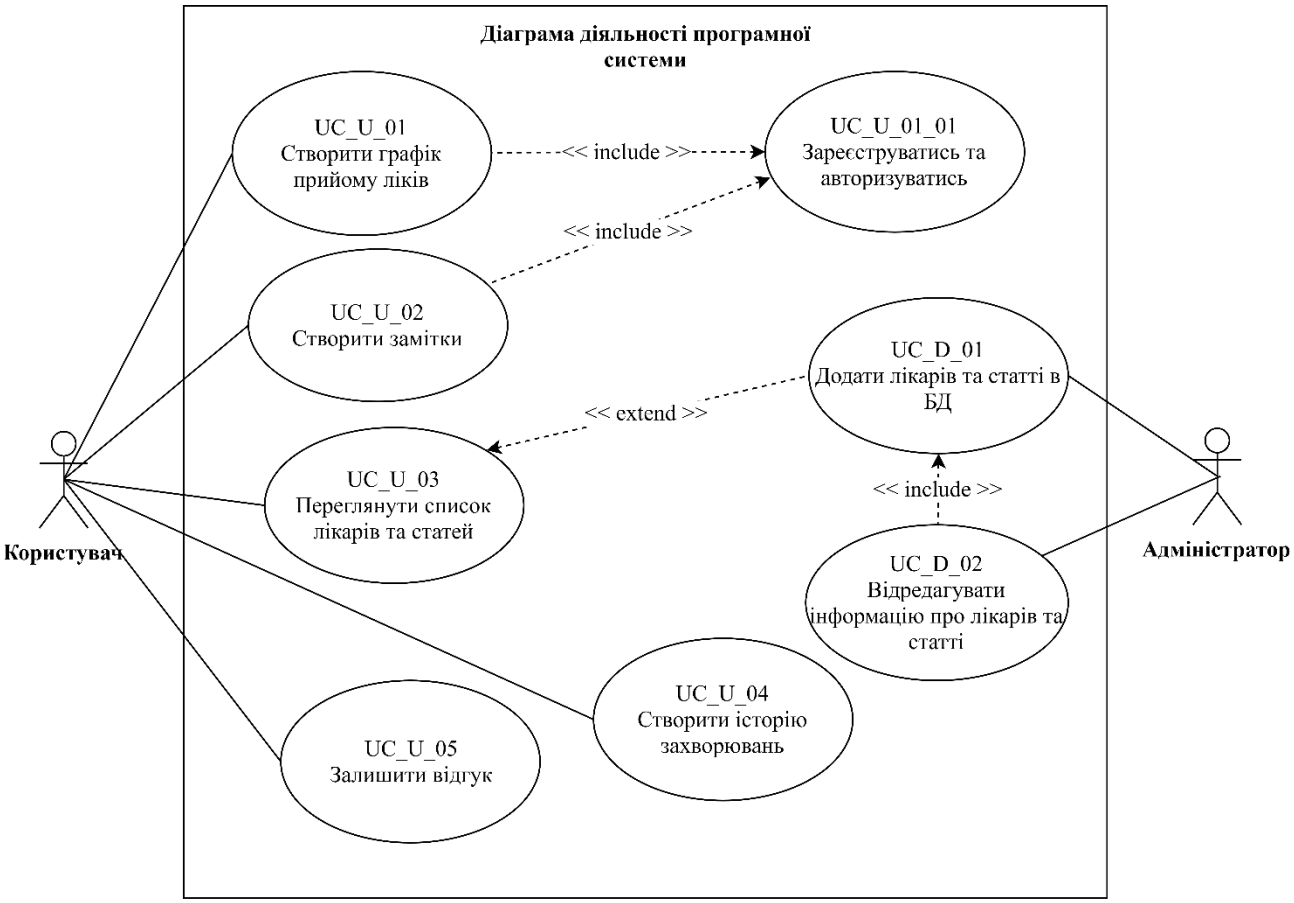
Копії графічних матеріалів



ДП.045440-06-99
Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги. Алгоритм взаємодії графічних елементів. Схема алгоритму.

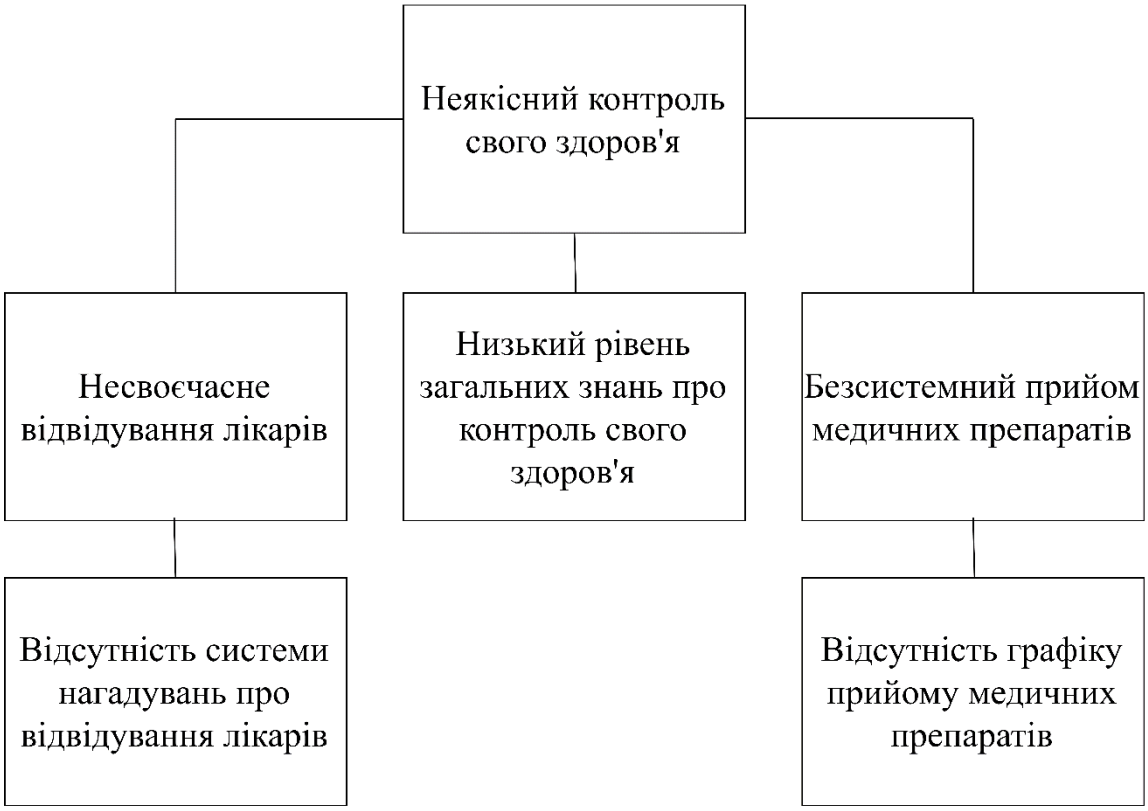


ДП.045440-07-99
Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги. База даних системи. Схема даних



Левошко Катерина, група КП-52

Дерево проблем



Левошко Катерина,група КП-52

Додаток 2

Лістинги програм

```
import UIKit

enum MyTheme {
    case light
    case dark
}
var theme = MyTheme.dark
class ViewController: UIViewController {



    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        self.noteTextView.text = ""
        if theme == .light {
            self.noteTextView.textColor = UIColor.black
        } else {
            self.noteTextView.textColor = UIColor.white
        }
    }
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        calenderView.changeTheme()
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        noteTextView.layer.borderColor = UIColor(red: 0.9, green: 0.9, blue:
0.9, alpha: 1.0).cgColor
        noteTextView.layer.borderWidth = 1.0
        noteTextView.layer.cornerRadius = 5
        self.title = "Календар"
        theme = .light
        Style.themeLight()
        self.navigationController?.navigationBar.isTranslucent=false
        self.view.backgroundColor=Style.bgColor

        view.addSubview(calenderView)
        calenderView.topAnchor.constraint(equalTo: view.topAnchor, constant:
10).isActive=true
        calenderView.rightAnchor.constraint(equalTo: view.rightAnchor,
constant: -12).isActive=true
```

```
        calenderView.leftAnchor.constraint(equalTo: view.leftAnchor,
constant: 12).isActive=true
        calenderView.heightAnchor.constraint(equalToConstant:
365).isActive=true

        let rightBarBtn = UIBarButtonItem(title: "Темний", style: .plain,
target: self, action: #selector(rightBarBtnAction))
        self.navigationItem.rightBarButtonItem = rightBarBtn

        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showProfile),
                                               name:
NSNotification.Name("ShowProfile"),
                                               object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showArticles),
                                               name:
NSNotification.Name("ShowArticles"),
                                               object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showDoctors),
                                               name:
NSNotification.Name("ShowDoctors"),
                                               object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector: #selector(share),
                                               name:
NSNotification.Name("Share"),
                                               object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showHistory),
                                               name:
NSNotification.Name("History"),
                                               object: nil)
        NotificationCenter.default.addObserver(self,
                                               selector:
#selector(showPills),
```

```
                                                name:
NSNotification.Name("ShowPills"),
                                                object: nil)
        NotificationCenter.default.addObserver(self,
                                                selector: #selector(rateUs),
                                                name:
NSNotification.Name("RateUs"),
                                                object: nil)
        NotificationCenter.default.addObserver(self,
                                                selector:
#selector(feedBack),
                                                name:
NSNotification.Name("FeedBack"),
                                                object: nil)
        NotificationCenter.default.addObserver(self,
                                                selector:
#selector(showAboutUs),
                                                name:
NSNotification.Name("AboutUs"),
                                                object: nil)
    }

    override func viewWillLayoutSubviews() {
        super.viewWillLayoutSubviews()

calenderView.myCollectionView.collectionViewLayout.invalidateLayout()
    }

    @objc func rightBarBtnAction(sender: UIBarButtonItem) {
        if theme == .dark {
            sender.title = "Темний"
            theme = .light
            Style.themeLight()
            self.noteTextView.textColor = UIColor.black
        } else {
            sender.title = "Світлий"
            theme = .dark
            Style.themeDark()
            self.noteTextView.textColor = UIColor.white
        }
        self.view.backgroundColor=Style.bgColor
        calenderView.changeTheme()
```

```
    }
    let calenderView: CalenderView = {
        let v=CalenderView(theme: MyTheme.dark)
        v.translatesAutoresizingMaskIntoConstraints=false
        return v
    }()

    @objc func showProfile() {
        performSegue(withIdentifier: "ShowProfile", sender: nil)
    }

    @objc func showSettings() {
        performSegue(withIdentifier: "ShowSettings", sender: nil)
    }

    @objc func showArticles() {
        performSegue(withIdentifier: "ShowArticles", sender: nil)
    }

    @objc func showDoctors() {
        performSegue(withIdentifier: "ShowDoctors", sender: nil)
    }

    @objc func share() {
        let activityController = UIActivityViewController(activityItems:
["App 'Doctor Help' developed by Kateryna Levoshko"], applicationActivities:
nil)
        self.present(activityController, animated: true)
    }

    @objc func showHistory() {
        performSegue(withIdentifier: "showHistory", sender: nil)
    }
    @objc func showPills() {
        performSegue(withIdentifier: "showPills", sender: nil)
    }
    @objc func rateUs() {
        let appstoreHooks = "itms-apps://itunes.apple.com/"
        let appstoreUrl = NSURL(string: appstoreHooks)
        if #available(iOS 10.0, *) {
            UIApplication.shared.open(appstoreUrl! as URL, options: [:],
completionHandler: nil)
```

```
            } else {
                UIApplication.shared.openURL(appstoreUrl! as URL)
            }
        }
        @objc func feedBack() {
            let email = "klevoshko98@gmail.com"
            if let url = URL(string: "mailto:\(email)") {
                if #available(iOS 10.0, *) {
                    UIApplication.shared.open(url)
                } else {
                    UIApplication.shared.openURL(url)
                }
            }
        }
        @objc func showAboutUs() {
            performSegue(withIdentifier: "showAboutUs", sender: nil)
        }

        @IBAction func onMoreTapped() {
            print("TOGGLE SIDE MENU")
            NotificationCenter.default.post(name:
    NSNotification.Name("ToggleSideMenu"), object: nil)
        }
        @IBOutlet weak var noteTextView: UITextView!
        override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
            if segue.identifier == "toNote" {
                if let destination = segue.destination as? NoteVC {
                    destination.cellFromCalendar = SavedNote.cell
                }
            }
            if segue.identifier == "toDoctorNote" {
                if let destination = segue.destination as? DoctorNoteVC {
                    destination.cellFromCalendar = SavedNote.cell
                }
            }
            if segue.identifier == "toPillNote" {
                if let destination = segue.destination as? PillNoteVC {
                    destination.cellFromCalendar = SavedNote.cell
                }
            }
        }
    }
```

```
import UIKit
import Firebase

class RegistrationVC: UIViewController, UITextFieldDelegate {

    let firebaseReference = Database.database().reference(withPath:
"users/")

    @IBOutlet weak var segmentedControl: UISegmentedControl!
    @IBOutlet weak var secondLabel: UILabel!
    @IBOutlet weak var thirdLabel: UILabel!
    @IBOutlet weak var nameTextField: UITextField!
    @IBOutlet weak var secondTextField: UITextField!
    @IBOutlet weak var thirdTextField: UITextField!
    @IBOutlet weak var emailTextField: UITextField!
    @IBOutlet weak var passTextField: UITextField!
    @IBOutlet weak var confirmTextField: UITextField!
    @IBOutlet weak var agreeButton: UIButton!
    @IBOutlet weak var signUpButton: UIButton!

    @IBAction func segmentedControlAction(_ sender: Any) {
        switch segmentedControl.selectedSegmentIndex {
        case 0:
            secondLabel.text = "Дата народження"
            thirdLabel.text = "Місто"
            break
        case 1:
            secondLabel.text = "Спеціалізація"
            thirdLabel.text = "Компанія"
            break
        default:
            break
        }
    }

    @IBAction func agreeButtonAction(_ sender: Any) {
        UIView.animate(withDuration: 0.2, delay: 0.0, options: .curveLinear,
animations: {
            self.agreeButton.transform = CGAffineTransform(scaleX: 0.1, y:
0.1)
        }) { (success) in
            self.agreeButton.isSelected = !self.agreeButton.isSelected
```

```
            UIView.animate(withDuration: 0.2, delay: 0.0, options:
.curveLinear, animations: {
                self.agreeButton.transform = .identity
                if self.agreeButton.isSelected &&
!self.nameTextField.text!.isEmpty &&
                    !self.secondTextField.text!.isEmpty &&
!self.thirdTextField.text!.isEmpty &&
                    !self.emailTextField.text!.isEmpty &&
!self.passTextField.text!.isEmpty &&
                    !self.confirmTextField.text!.isEmpty &&
                    self.passTextField.text == self.confirmTextField.text {
                    self.signUpButton.isEnabled = true
                } else {
                    self.signUpButton.isEnabled = false
                }
            }, completion: nil)
        }

    }

    @IBAction func signUpButtonAction(_ sender: Any) {
        Auth.auth().createUser(withEmail: emailTextField.text!, password:
passTextField.text!) { user, error in
            if error == nil {
                print("SUCCESS")


                let newUser = User(name: self.nameTextField.text!,
                                   db: self.secondTextField.text!,
                                   city: self.thirdTextField.text!,
                                   email: self.emailTextField.text!,
                                   weight: "-",
                                   height: "-",
                                   telephone: "-")

                let newUserReference =
self.firebaseReference.child("\(self.emailTextField.text!.replacingOccurrenc
es(of: ".", with: ","))")
                newUserReference.setValue(newUser.toAnyObject())
```

```
                let userEmail =
self.emailTextField.text!.replacingOccurrences(of: ".", with: ",")
                UserDefaults.standard.set(userEmail, forKey: "user")
                self.performSegue(withIdentifier: "fromSignupToCalendar",
sender: self)
            } else {
                var errorText = ""
                if error?.localizedDescription == "The email address is
badly formatted." {
                    errorText = "Неправильний формат вводу електронної
пошти."
                } else if error?.localizedDescription == "The email address
is already in use by another account." {
                    errorText = "Такий e-mail вже використовується іншим
користувачем"
                }
                let alert = UIAlertController(title: "Помилка", message:
errorText, preferredStyle: .alert)
                print(error?.localizedDescription)
                let alertAction = UIAlertAction(title: "Ввести ще раз",
style: .default, handler: nil)
                alert.addAction(alertAction)
                self.present(alert, animated: true, completion: nil)
            }
        }

    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        self.navigationController?.isNavigationBarHidden = false
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        nameTextField.delegate = self
        secondTextField.delegate = self
        thirdTextField.delegate = self
        emailTextField.delegate = self
        passTextField.delegate = self
        confirmTextField.delegate = self
```

```
        self.navigationController?.isNavigationBarHidden = false
        secondLabel.text = "Дата народження"
        thirdLabel.text = "Місто"

        passTextField.isSecureTextEntry = true
        confirmTextField.isSecureTextEntry = true


        [nameTextField, secondTextField, thirdTextField, emailTextField,
passTextField, confirmTextField].forEach({ $0.addTarget(self, action:
#selector(editingChanged), for: .editingChanged) })
    }

    @objc func editingChanged(_ textField: UITextField) {
        if textField.text?.characters.count == 1 {
            if textField.text?.characters.first == " " {
                textField.text = ""
                return
            }
        }
        guard
            let _ = nameTextField.text, !nameTextField.text!.isEmpty,
            let _ = secondTextField.text, !secondTextField.text!.isEmpty,
            let _ = thirdTextField.text, !thirdTextField.text!.isEmpty,
            let _ = emailTextField.text, !emailTextField.text!.isEmpty,
            let _ = passTextField.text, !passTextField.text!.isEmpty,
            let _ = confirmTextField.text, !confirmTextField.text!.isEmpty,
            agreeButton.isSelected, confirmTextField.text ==
passTextField.text
            else {
                signUpButton.isEnabled = false
                return
        }
        signUpButton.isEnabled = true
    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }
```

```
    func textField(_ textField: UITextField, shouldChangeCharactersIn range:
NSRange, replacementString string: String) -> Bool {

        if(string == "\n") {
            textField.resignFirstResponder()
            return false
        }
        if textField == secondTextField &&
segmentedControl.selectedSegmentIndex == 0 {
            let aSet = NSCharacterSet(charactersIn:"1234567890.").inverted
            let compSepByCharInSet = string.components(separatedBy: aSet)
            let numberFiltered = compSepByCharInSet.joined(separator: "")
            return string == numberFiltered
        }
        return true
    }
}
```

Додаток З

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНА СИСТЕМА ПЛАНУВАННЯ МЕДИЧНОГО ОБСЛУГОВУВАННЯ У ЛІКАРІВ ПЕРВИННОЇ ЛАНКИ МЕДИЧНОЇ ДОПОМОГИ

Виконала: Левошко Катерина Василівна

Науковий керівник: старший викладач кафедри ПЗКС, к.т.н., Люшенко Л.А.

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмний додаток планування медичного обслуговування.

Завдання:

1. Проаналізувати предметну область, визначити її актуальність та проблематику. А також існуючі програмні рішення та визначити вимоги до розроблення програмного додатка.
2. Розробити програмний додаток згідно вимог за допомогою обраних засобів реалізації.
3. Протестувати розроблений програмний додаток та проаналізувати його відповідність до вимог.

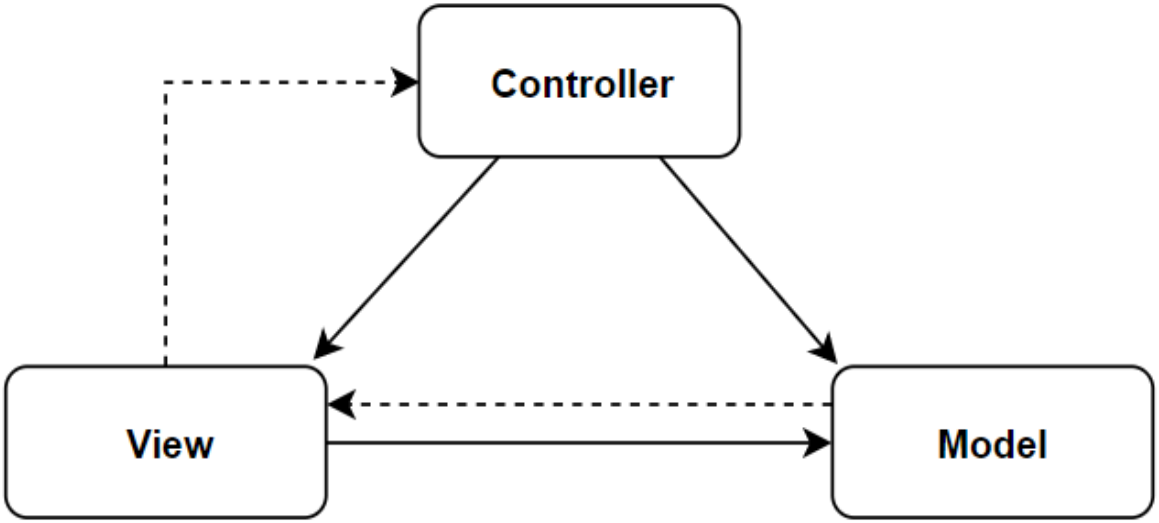
АКТУАЛЬНІСТЬ

Медична сфера активно розвивається в цифровому світі сьогодні. В стані повного інформаційного навантаження сучасна людина потребує систему для контролю медичного обслуговування. Програмне забезпечення повинно бути доступним, тому мобільний додаток – найбільш оптимальне рішення.

Тож, розробка програмного забезпечення у сфері медицини є актуальним рішенням сьогодні.

АРХІТЕКТУРА СИСТЕМИ

Архітектура програмної системи створена на основі шаблону проектування MVC.



АРХІТЕКТУРА СИСТЕМИ

Модель – база даних із сутностями та зв’язками між ними.

Представлення – екранні форми мобільного додатка.

Контроллер – управляє передачею даних від моделі до представлення.

Архітектура системи гнучка завдяки шаблону проектування MVC. В результаті – система є надійною та здатною до масштабування.

ЗАСОБИ РОЗРОБЛЕННЯ

Платформа – iOS:

- гнучке середовище розробки Xcode, потужний інструмент тестування та симуляції;
- централізованість платформи, завдяки якій синхронізація у мобільному додатку може бути виконана набагато легше та безпечніше;
- висока платоспроможність користувачів iOS, що означає перспективу розвитку додатка в цілях монетизації.

ЗАСОБИ РОЗРОБЛЕННЯ

Мова програмування – Swift:

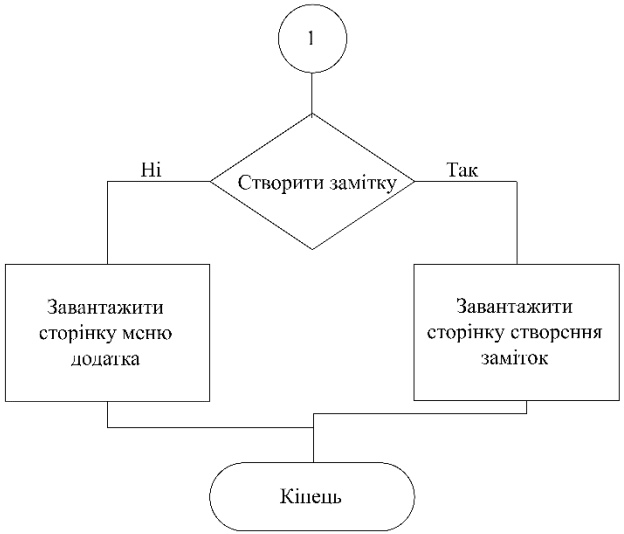
- якісна та регулярна підтримка від компанії Apple (в порівнянні з Objective-C, яка вже застаріла і майже не підтримується);
- швидка та зручна робота з програмним кодом Swift. Його набагато простіше читати розробнику системи, а також модифікувати та створювати;
- використання швидкого компілятора – в порівнянні з іншими інтерпретованими мовами PHP і JS;

ЗАСОБИ РОЗРОБЛЕННЯ

База даних – Firebase:

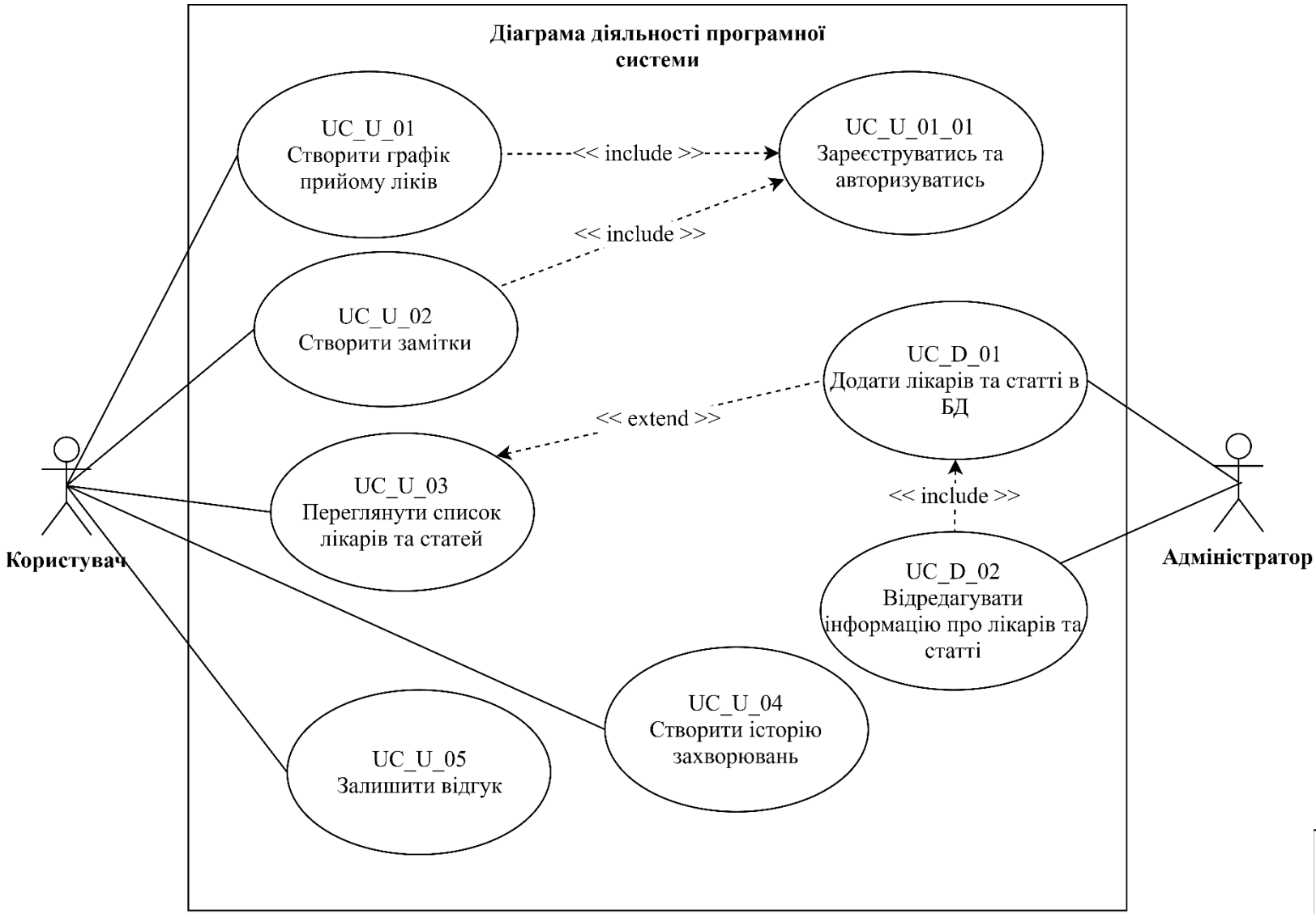
- швидкодія роботи з даними. База даних реалізована з використанням технології «Хмарне сховище»;
- готові шаблони способів реєстрації та авторизації;
- простота в роботі з системою. Розробнику легко працювати з базою даних – підключити її, додавати та видаляти записи;
- активна підтримка від Google та безкоштовний зворотній зв’язок від спеціалістів по розробці.

РОЗРОБЛЕНІ АЛГОРИТМИ



ДП.045440-06-99
Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги. Алгоритм взаємодії графічних елементів. Схема алгоритму.

Діаграма діяльності програмної системи



КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ

Відповідність розробленим вимогам до ПЗ:

- реєстрація в системі (за допомогою e-mail, тобто є зв'язок з електронною скринькою);
- авторизація в системі за допомогою e-mail та паролю;
- створення графіку прийому лікарських засобів;
- додавання нагадування прийому лікарських засобів;
- додавання нагадування прийому у лікаря;
- додавання текстових заміток у календарі;
- редагування даних в особистому кабінеті;
- перегляд тематичних медичних статей;

КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ

Відповідність розробленим вимогам до ПЗ:

- перегляд списку лікарів та інформації про них;
- додавання історії захворювань та супутньої інформації;
- можливість залишити відгук;
- можливість прочитати інформацію про розроблену систему;

1) наявність користувацької згоди на обробку даних на правил користування додатком;

2) обробка додатком помилок користувача.

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

У процесі тестування методом «чорного ящика» було виявлено кілька незначних помилок у створені та взаємодії інтерфейсу:

- було знайдено граматичні помилки у тексті повідомлень;
- критична помилка при введені спеціальних значень у форми мобільного додатку

Перша помилка була зроблена через людський фактор, яку неможливо було відслідкувати під час автоматичного тестування. Друга помилка була пов’язана з особливістю реалізації і після коректування реалізації ця проблема зникла.

ПОРІВНЯННЯ З АНАЛОГАМИ

Назва	Інтерфейс	Наявність функцій нагадування	Використання в межах України	Наявність користувацької згоди	Використання хмарних технологій
«Моё здоров’я: образ жизни»	Створено без вимог	Так	Ні	Ні	Так
«Дневник здоров’я»	Створено без вимог	Ні	Так	Ні	Ні
«Youwell»	Створено без вимог	Ні	Так	Ні	Ні

РЕКОМЕНДАЦІЇ ДЛЯ ПОКРАЩЕННЯ

- реалізація ролі користувача «Лікар» та взаємодію лікарів та пацієнтів;
- можливість онлайн зв’язку лікарів та пацієнтів через чат або відео-зв’язок;
- можливість реєструватись та авторизуватись за допомогою Google, Facebook;
- додання можливості пошуку та сортування лікарів по спеціалізації, стажу, вартості прийому;
- можливість записатись автоматично на прийом до лікаря та налаштувати синхронізацію онлайн-запису та нагадувань про запис;
- можливість робити покупки в мобільному додатку (оплачувати онлайн-консультацію лікаря).

ВИСНОВКИ

1. Був проведений аналіз існуючих програмних рішень.
2. Запропоновано архітектуру ПЗ на основі шаблону проектування MVC.
3. Були визначені функціональні вимоги до розроблюваної системи, серед яких є апаратні вимоги, вимоги до інтерфейсу, вимоги до технологій.
4. Розроблено ПЗ, яке задовольняє критерії якості ПЗ.
5. Протестовано та порівняно ПЗ з аналогами за критеріями, які були визначені при аналізі існуючих рішень (переваги на недоліки).
6. Мобільний додаток розроблений в повному обсязі з відповідністю до розроблених вимог. Тестування мобільного додатка виконано в повному обсязі згідно наявної методики тестування.
7. Розроблений мобільний додаток допомагає користувачам економити свій час та створити систему власного медичного обслуговування у своєму смартфоні.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНА СИСТЕМА ПЛАНУВАННЯ МЕДИЧНОГО ОБСЛУГОВУВАННЯ У ЛІКАРІВ ПЕРВИННОЇ ЛАНКИ МЕДИЧНОЇ ДОПОМОГИ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ К.В.Левошко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги являє собою мобільний додаток, створений на платформі iOS (мова програмування Swift) з використанням шаблону проектування MVC.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) відповідність елементів сторінок вимогам до мобільного додатку;
2) наявність доступу до календаря та тематичних статей;
4) наявність обробки критичних ситуацій;
5) зручність роботи з мобільним додатком;
6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Black Box Testing. Перевіряється безпосередньо програмний продукт на відповідність функціональним вимогам з точки зору зовнішнього світу, без урахування знань про внутрішнє представлення системи. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування граничних значень;
4) тестування за допомогою припущення про помилку;
5) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію Appium.

Працездатність програмної системи планування медичного обслуговування перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування мобільного додатка на пристроях різних моделей (з різними розмірами та апаратним забезпеченням);
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ПЛАНУВАННЯ МЕДИЧНОГО ОБСЛУГОВУВАННЯ У ЛІКАРІВ ПЕРВИННОЇ ЛАНКИ МЕДИЧНОЇ ДОПОМОГИ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ К.В. Левошко

2019

ЗМІСТ

1. Опис структури мобільного додатка

Програмна система планування медичного обслуговування у лікарів первинної ланки медичної допомоги складається із статичних сторінок та сторінок, вміст яких формується динамічно. Мобільний додаток виконаний українською мовою.

Статичні сторінки мобільного додатка:

– початкова сторінка (при запуску додатка) має чотири екрани, перехід по яким можна робити за допомогою слайдера;
– сторінка с вибором входу або реєстрації користувача;
– меню мобільного додатка;
– особистий профіль користувача;
– сторінка лікарів;
– тематичні статті;
– сторінки для відгуку;
– інформація про додаток;
– сторінка з правилами використання мобільного додатка;
– сторінка з користувацькою згодою.

Динамічні сторінки мобільного додатка:

– сторінка реєстрації користувача;
– сторінка входу користувача;
– календар;
– сторінка створення текстової замітки;
– сторінка додавання нагадування прийому у лікаря;
– сторінка додавання прийому ліків;
– сторінка створення ліків;
– історія захворювань.

Кожна сторінка містить посилання минулу сторінку мобільного додатка або наявні елементи, за допомогою яких користувач перейде до наступної сторінки.

2. Опис вмісту статичних сторінок мобільного додатка

Початкова сторінка додатка.

Містить чотири екрани, які з'єднані одне з одним за допомогою слайдера. Перші три екрани коротко описують сервіси, які надає мобільний додаток та показують назву (див. рис. 1, рис. 2, рис. 3).

Рис. 1. Перший екран початкової сторінки додатка

Рис. 2. Другий екран початкової сторінки додатка

Рис. 3. Третій екран початкової сторінки додатка

Четвертий екран містить логотип мобільного додатка та дві кнопки «Увійти» та «Зареєструватись» (див. рис. 4).

Рис. 4. Четвертий екран початкової сторінки додатка

Меню додатка.

До меню мобільного додатка можна доступитись, якщо провести вправо по екрану смартфона. Меню складається з дев’яти пунктів і воно поділено на дві частини – сервіси для користувача та для розробників (див. рис. 5). Перша частина містить сторінки «Мій профіль», «Лікарі», «Статті», «Ліки», «Історія захворювань». Друга частина містить сторінки «Поділитись з друзями», «Оцініть нас», «Зворотній зв’язок», «Про додаток».

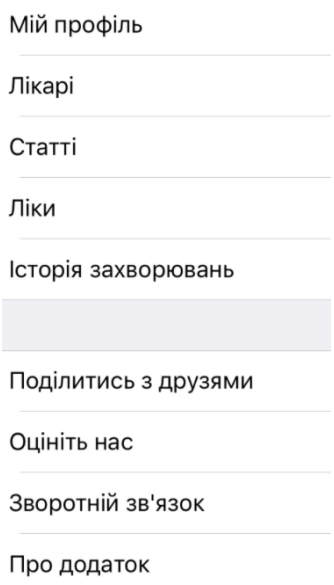


Рис. 5. Меню мобільного додатка

Особистий профіль користувача.

Ця сторінка містить інформацію, яку користувач вводив при реєстрації, а також додаткові поля – телефон, зріст, вага. Користувач може вносити зміни та зберігати їх, а також вийти із системи (див. рис. 6).

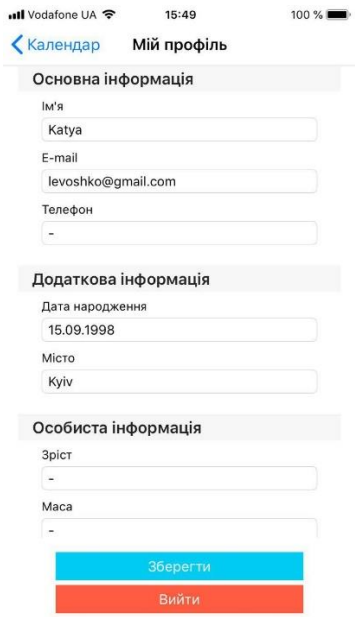


Рис. 6. Сторінка особистого профілю користувача

Сторінка лікарів.

Ця сторінка показує лікарів та інформацію про них – ім’я, спеціалізацію, вартість прийому, фотографію (див. рис. 7).

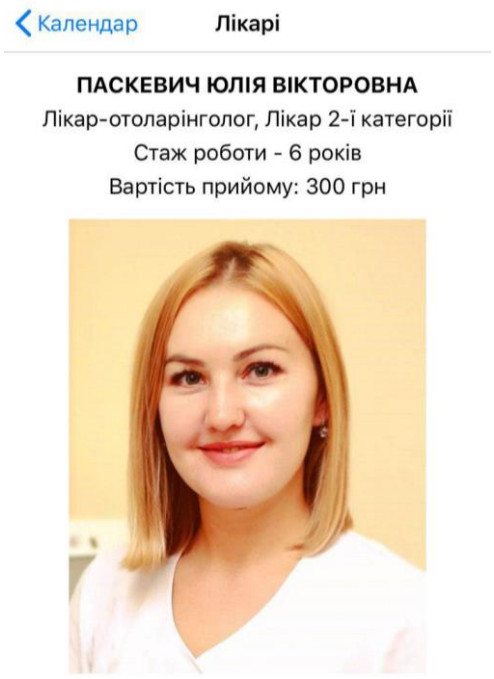


Рис. 7. Сторінка зі списком лікарів

Тематичні статті.

Ця сторінка показує тематичні статті і містить заголовок та картинку до статті (див. рис. 8). Користувач може натиснути на картинку або заголовок та перейти до прочитання самої статті (див. рис. 9).

Рис. 8. Сторінка зі списком тематичних статей

Рис. 9. Сторінка статті «Вправи для зору при короткозорості»

Сторінки для відгуку.

До цих сторінок відносяться пункти меню «Поділитись з друзями», «Оцініть нас», «Зворотній зв’язок». При натисканні «Поділитись з друзями» відкривається стандартна панель iPhone при відправці посилання на додаток через різні програмні засоби (див. рис. 10). При натисканні «Оцініть нас» відкривається Appstore, на якому користувач може поставити оцінку мобільному додатку. При натисканні «Зворотній зв’язок» відкривається додаток електронної пошти та користувач може написати розробнику e-mail з побажаннями та рекомендаціями для покращення мобільного додатку.

Рис. 10. Сторінка «Поділитись з друзями»

Інформація про додаток.

На цій сторінці міститься необхідна інформація про розроблений додаток – назва, версія, ім’я та контакти розробника та загальна інформація про розроблений програмний додаток (див. рис. 11). А також тут міститься посилання на користувацьку згоду.

Рис. 11. Сторінка «Інформація про додаток»

Сторінка з користувацькою згодою.

На цій сторінці (див. рис. 12) міститься користувацька згода в текстовому форматі.

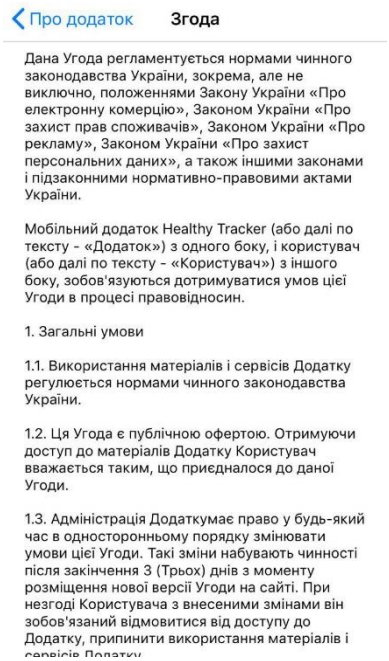


Рис. 12. Сторінка з користувацькою згодою

Сторінка з правилами використання мобільного додатка.

На цій сторінці (див. рис. 13) правила використання мобільного додатка в текстовому форматі. При реєстрації користувач має можливість прочитати правила, в яких э згода на обробку персональних даних.

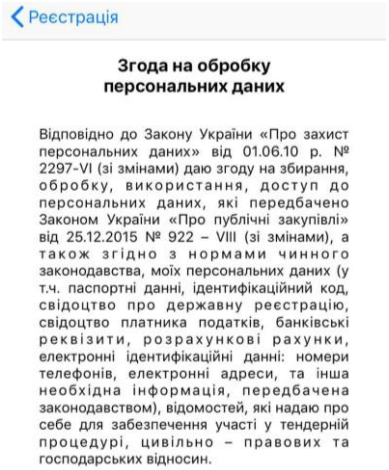


Рис. 13. Сторінка з правилами використання мобільного додатка

3. Процедура реєстрації та авторизації користувача

Процедура реєстрації.

При натисканні кнопки «Зареєструватись» (див. рис. 4), користувач переходить до сторінки реєстрації (див. рис. 14). Зверху можна обрати яку роль користувача реєструвати «Пацієнт» або «Лікар». У розробленій версії мобільного додатка реалізовано тільки роль користувача «Пацієнт». Для того щоб зареєструватись користувач має ввести наступні дані – ім’я, дату народження, місто, e-mail, пароль та повторити пароль. Введення даних до всіх полів оброблено згідно необхідної валідації. Наприклад, якщо користувач з введеним e-mail вже існує, то виникає помилка (див. рис. 15).

Для реєстрації користувач повинен прочитати правила використання мобільного додатка (див. рис. 13) та натиснути на «галочку» біля

посилання на правила (див. рис. 14). Після процедури реєстрації користувач автоматично авторизується в системі.

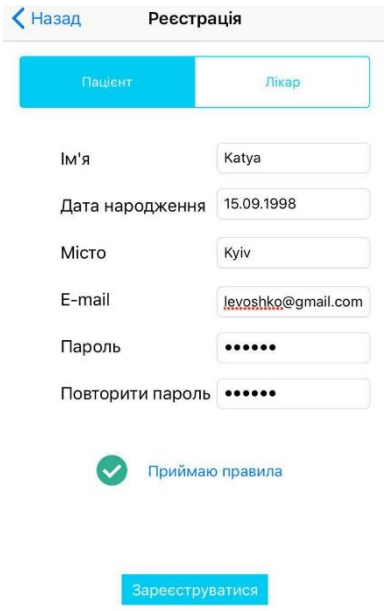


Рис. 14. Сторінка реєстрації користувача

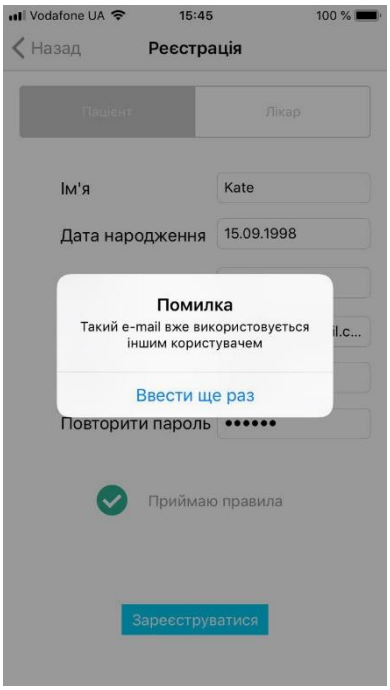


Рис. 15. Повідомлення про помилку на сторінці реєстрації

Процедура авторизації.

При натисканні кнопки «Увійти» (див. рис. 4), користувач переходить до сторінки авторизації (див. рис. 16). Для того, щоб авторизуватись у системі користувач повинен ввести e-mail та пароль, який він вводив при реєстрації. Введення даних до всіх полів оброблено згідно необхідної валідації. Наприклад, користувача с таким e-mail не знайдено, то виникає помилка (див. рис. 17). Помилки оброблено аналогічно до полів на сторінці «Реєстрація».

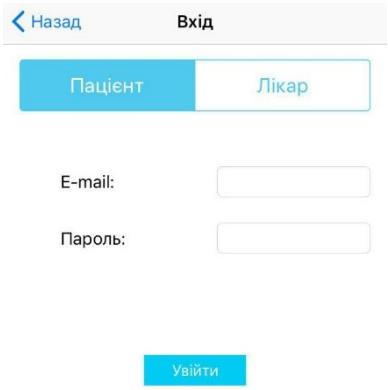


Рис. 16. Сторінка авторизації користувача

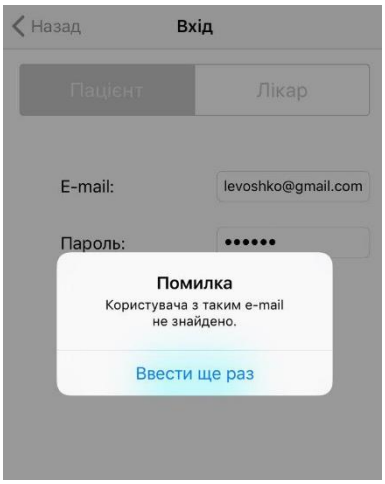


Рис. 17. Повідомлення про помилку на сторінці авторизації

4. Процедура додавання ліків та історії захворювань

Додавання ліків.

Для того, щоб створювати графік прийому ліків у мобільному додатку треба додати ліки. Для цього у меню треба перейти до пункту «Ліки» (див. рис. 18). Якщо користувач ще не додавав ліки, то він побачить повідомлення, що в нього ще немає доданих ліків та у верхньому правому куті буде знак «+» для додавання ліків.

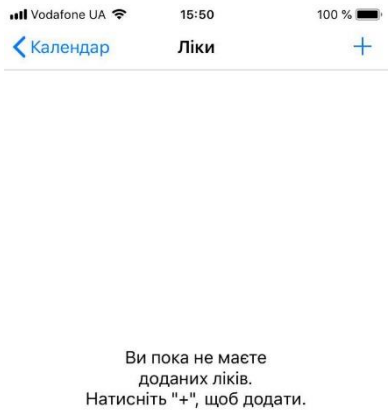


Рис. 18. Сторінка «Ліки» у меню додатка

На сторінці створення ліків (див. рис. 19) є наступні поля – назва ліків, тип ліків (можна обрати з випадаючого списку таблетки, мазь, сироп, краплі та інший тип), кількість разів прийому в день, кількість днів, опис ліків та кнопка «Додати». Після додавання ліків до списку вони будуть представлені у загальному списку (див. рис. 20).

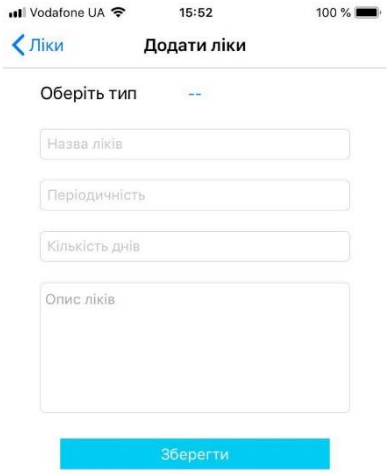


Рис. 19. Сторінка додавання ліків до списку

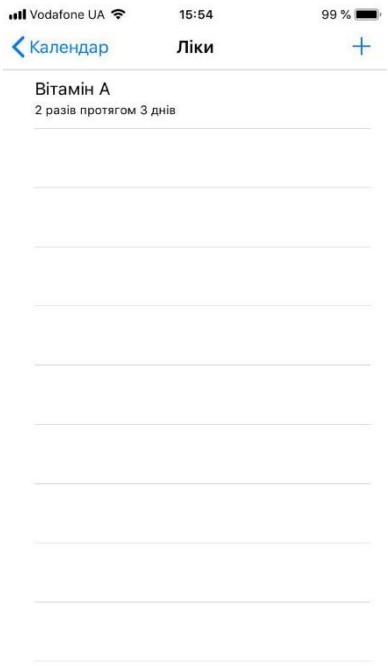


Рис. 20. Сторінка «Ліки» після додавання ліків до списку

Додавання історії захворювань.

Для того, щоб додати історію захворювань треба перейти до пункту «Історія захворювань» у меню додатка (див. рис. 21). Якщо користувач ще не додавав історію захворювання, то він побачить повідомлення, що в нього ще немає доданих захворювань та у верхньому правому куті буде знак «+» для їх додавання.

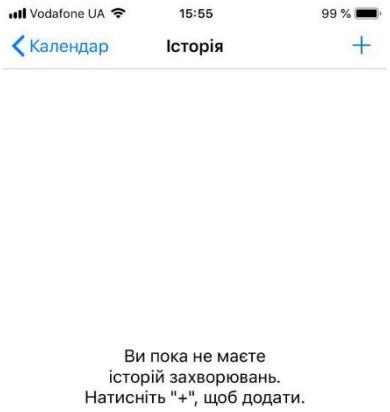


Рис. 21. Сторінка «Історія захворювань» у меню додатка

На сторінці створення захворювання (див. рис. 22) є наступні поля – назва захворювання, період захворювання, опис захворювання та кнопка «Додати». Після додавання захворювань до списку вони будуть представлені у загальному списку (див. рис. 20).

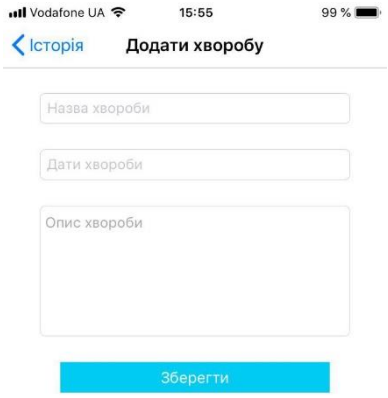


Рис. 22. Сторінка додавання захворювань до списку

Рис. 23. Сторінка «Історія захворювань» після додавання захворювань до списку

5. Процедура взаємодії з календарем

Після реєстрації або авторизації користувач переходить до сторінки календаря (див. рис. 24), який має світлий фон за замовчуванням. У верхньому правому куті можна змінити фон на темний (див. рис. 25).

Рис. 24. Календар мобільного додатка (світлий фон)

Рис. 25. Календар мобільного додатка (темний фон)

При натисканні на будь-яку дату календаря двічі користувачеві доступні чотири дії: створити нагадування про прийом ліків, створити нагадування про прийом у лікаря, створити текстову замітку та відмінити цю дію (див. рис. 26).

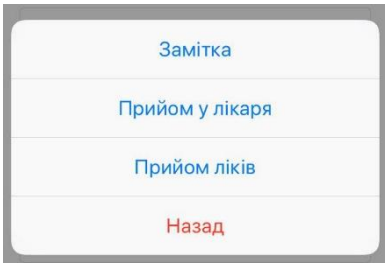


Рис. 26. Список дій при натисканні на дату календаря

При натисканні на пункт списку «Прийом ліків» відкривається вікно створення графіку прийому ліків (див. рис. 27). Там можна обрати з випадаючого списку вже створені ліки на сторінці «Ліки» та додатку замітку, яку користувач буде бачити при натисканні на дату в календарі.

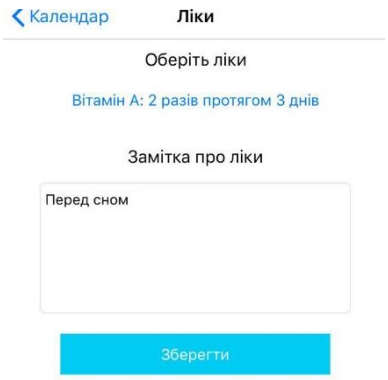


Рис. 27. Сторінка створення графіку прийому ліків

При натисканні на пункт списку «Прийом у лікаря» відкривається вікно створення нагадування прийому у лікаря (див. рис. 28). Там можна обрати з випадаючого списку тип лікаря (спеціалізацію), лікаря, обрати час

через зручний слайдер часу та додатку замітку, яку користувач буде бачити при натисканні на дату в календарі.

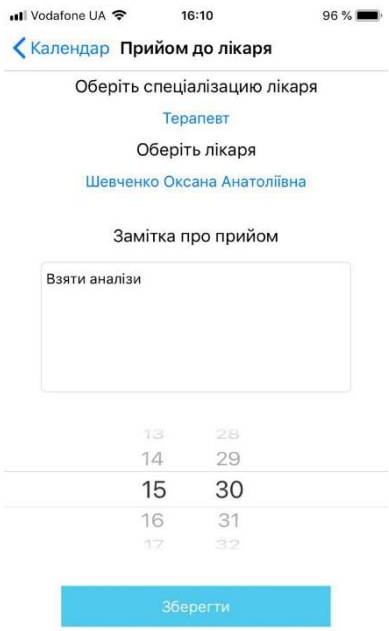


Рис. 28. Сторінка створення нагадування прийому у лікаря

При натисканні на пункт списку «Замітка» відкривається вікно створення замітки (див. рис. 29). Там можна додатку замітку, яку користувач буде бачити при натисканні на дату в календарі.

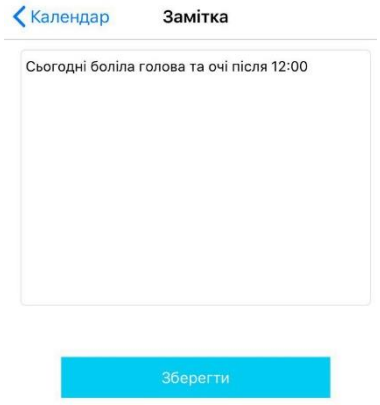


Рис. 29. Сторінка створення текстової замітки

Після створення всіх необхідних заміток та нагадувань, дати календаря, в якому є замітки виділяються зеленим кольором, а обрана користувачем дата виділяється блакитним кольором та знизу показується вся необхідна інформація про замітки (див. рис. 30).

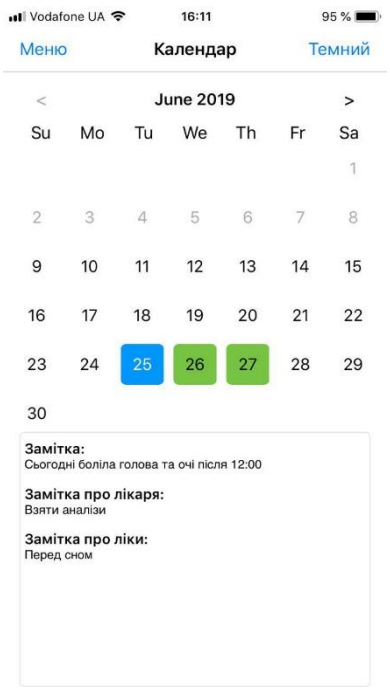


Рис. 30. Календар після додавання заміток